A TRIDENT SCHOLAR PROJECT REPORT

NO. 439

Multiple-Sensor Discrimination of Closely-Spaced Objects on a Ballistic Trajectory

by

Midshipman 1/C Samuel S. Lacinski, USN

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

USNA-1531-2

U.S.N.A. --- Trident Scholar project report; no. 439 (2015)

Multiple-Sensor Discrimination of Closely-Spaced Objects on a Ballistic Trajectory

by

Midshipman 1/C Samuel S. Lacinski, USN
United States Naval Academy
Annapolis, Maryland

_____
(signature)

Certification of Adviser(s) Approval

Associate Professor Tae W. Lim
Aerospace Engineering Department

_____
(signature)

_____
(date)

Commander Tracie A. Severson, USN
Systems Engineering Department

_____
(signature)

_____

(date)

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

_____
(signature)

_____
(date)

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
05-18-2015		

4. TITLE AND SUBTITLE
Multiple-Sensor Discrimination of Closely-Spaced Objects on a Ballistic Trajectory

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Lacinski, Samuel S.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Naval Academy
Annapolis, MD 21402

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Trident Scholar Report no. 439 (2015)

12. DISTRIBUTION / AVAILABILITY STATEMENT

This document has been approved for public release; its distribution is UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

One of the challenges associated with defending against ballistic missiles is to discriminate the object of interest among multiple closely-spaced objects (CSOs) that travel on a ballistic trajectory. The discrimination process typically involves the identification and tracking of the object of interest. One approach that can improve discrimination performance is to employ multiple sensors. Multiple-sensor correlation and discrimination involves the integration of sensor measurements collected from terrestrial and on-orbit sensors to improve the likelihood of identifying and tracking an object of interest within the CSOs. This report describes the development of the algorithms necessary for fusing sensor measurement data obtained from multiple, dissimilar sensors in order to improve the likelihood of identifying and tracking an object of interest within closely-spaced objects traveling on a ballistic trajectory. The algorithms utilize a target object map (TOM) that is created using multiple sensor measurements for correlation. The object of interest is then selected using a probability-based Dempster-Shafer discrimination algorithm. To examine the performance of these algorithms, a simulation environment was developed. It included relevant characteristics of the sensors in the discrimination system, a modeling process for the ballistic trajectories of the CSOs, and a decision-making module containing the algorithms for handling the sensor returns and correlating and discriminating the object of interest.

15. SUBJECT TERMS
closely-spaced ballistic objects, dissimilar sensor correlation, target object map, Dempster-Shafer discrimination algorithms, simulation environment for various operational scenarios, sensitivity analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		64	**19b. TELEPHONE NUMBER** *(include area code)*

Abstract

One of the challenges associated with defending against ballistic missiles is to discriminate the object of interest among multiple closely-spaced objects (CSOs) that travel on a ballistic trajectory. The discrimination process typically involves the identification and tracking of the object of interest. One approach that can improve discrimination performance is to employ multiple sensors. Multiple-sensor correlation and discrimination involves the integration of sensor measurements collected from terrestrial and on-orbit sensors to improve the likelihood of identifying and tracking an object of interest within the CSOs. This report describes the development of the algorithms necessary for fusing sensor measurement data obtained from multiple, dissimilar sensors in order to improve the likelihood of identifying and tracking an object of interest within closely-spaced objects traveling on a ballistic trajectory. The algorithms utilize a target object map (TOM) that is created using multiple sensor measurements for correlation. The object of interest is then selected using a probability-based Dempster-Shafer discrimination algorithm. To examine the performance of these algorithms, a simulation environment was developed. It included relevant characteristics of the sensors in the discrimination system, a modeling process for the ballistic trajectories of the CSOs, and a decision-making module containing the algorithms for handling the sensor returns and correlating and discriminating the object of interest. This simulation environment was iterated to assess the effect of varying radar angular and range resolution and discrimination characteristic distribution overlap on discrimination performance. This project found there was not a strong correlation between changing radar resolution by +/-10% from the nominal values and probability of discrimination. However, it was noted that a 35 percent increase in track uncertainty for an increase of 0.1 degrees in angular resolution and a six percent increase in zero effort miss distance for the same change in angular resolution. This project also noted a strong inverse quadratic relationship between the characteristic overlap, which is determined by the amount of overlap in the probability of observation for different classes of objects, and discrimination performance.

Keywords: (1) closely-spaced ballistic objects, (2) multiple, dissimilar sensor correlation, (3) target object map, (4) Dempster-Shafer discrimination algorithms, (5) simulation environment for various operational scenarios, (6) sensitivity analysis

Acknowledgements

This project would not be possible if it were not for a few incredible individuals. The Group 31 staff members at MIT-Lincoln Laboratory were tremendous hosts for two summer internships that were instrumental in my interest in this topic. In particular, Dr. Thierry Copie, Senior Staff in Group 31, has been a mentor throughout the process, from my first day interning with his group until the completion of this research. His patience and technical knowledge were essential ingredients in this process. Dr. Sami Amasha, Technical Staff in Group 31, enabled my theoretical understanding and provided the MATLAB implementation of the Dempster-Shafer algorithms utilized in this paper. LCDR Barry Willhite, USNR, formerly of the US Naval Academy Aerospace Engineering Department, helped guide the initial stages of this project. Finally, Professor Tae Lim, US Naval Academy Aerospace Engineering Department and CDR Tracie Severson, US Naval Academy Systems Engineering Department, have served as advisors, co-authors, and mentors throughout this research process. Without their dedication to this project and their insightful guidance, this research would have not been as fruitful.

Preface

Ballistic missiles have long been a challenging tactical, operational, and strategic asset as well as problem for militaries around the world. The strategic stability of the Cold War was largely predicated on the invulnerability of nuclear-tipped land and submarine launched intercontinental ballistic missiles.[1] However, as the technologies for these systems have matriculated from major strategic powers to smaller, less predictable rogue states, a new tactical and strategic threat has emerged.[2] The performance of the Patriot system in the Gulf War, shooting down 51 of 88 potential threats at a cost of 157 expended interceptors[3], highlighted the need for better missile defense systems to protect theater forces from ballistic missile threats, while the possible threat of a rogue missile attack on the United States homeland motivates a system capable of intercepting intercontinental threats[4]. These systems are necessary not merely to defend America and its allies from potential attack but render ineffective the threat of potential attack. In this manner, "ballistic missile defense is not simply a *shield* but an *enabler* of U.S. action."[5] To fail to develop flexible, effective missile defense systems would be to surrender the strategic imperative around the world, a fate that should be avoided by the United States.

Table of Contents

List of Figures

List of Tables

List of Symbols and Acronyms

BMD .....Ballistic Missile Defense
CSO.....Closely Spaced Object
DCM… .....Direction Cosine Matrix
b_x... .....measurement bias
$\mathbf{C_x}$ .....state estimate covariance matrix
D.....direction cosine matrix
EM.....Evidence Measure
F .....system dynamics matrix
FOM.....Figure of Merit
H .....Measurement matrix
KF .....Kalman Filter
$\mathbf{K_k}$.....Kalman gain matrix
KV.....Kill Vehicle
$\mathbf{M_k}$ .....covariance matrix prior to update
$\underline{P}$.....pointing vector from kill vehicle to re-entry vehicle
P_c.....probability of correlation
P_d.....probability of discrimination
$\mathbf{P_k}$.....covariance matrix after update
PP .....Pignistic Probability
$\mathbf{Q_k}$.....process noise matrix
r.....correlation coefficient
$\underline{R_i}$ .....position of the kill vehicle
$\mathbf{R_k}$ .....discrete measurement matrix
$\underline{R_t}$ .....position of the re-entry vehicle
RV.....re-entry vehicle
TOM.....Target Object Map
X1.....Nominal X-band radar
w_k.....discrete noise
x_k.....truth measurement
$\widehat{\boldsymbol{x}}_{\boldsymbol{k}}$ .....estimated state vector
$\overline{\boldsymbol{x}}_{\boldsymbol{k}}$ .....previous estimated state vector
$\mathbf{z_k}$ .....measurement vector
ZEM… .....Zero Effort Miss Distance
θ.....uncertainty
ϕ .....conflict
σ.....variance

1 Introduction

Discrimination, at its heart, is nothing more than decision making. It is defined as the ability to make fine distinctions. Discrimination in a technical context is then the synthesis of the given information in a coherent manner in order to "make a fine distinction." Various factors convolute and confuse this process. For example, the information utilized for making this decision is almost always imperfect because of sensor error, and in many cases information from different sources can support different decisions. In some scenarios such as military applications, the situation often demands dynamic and responsive decisions due to challenging environments. In these situations a wrong decision may lead to dire consequences.

While discrimination in general has a variety of applications, this project is focused on discrimination in the context of Ballistic Missile Defense (BMD). In a typical BMD system, a set of sensors is responsible for observing a complex of objects, selecting a target to engage, and intercepting that target.[6] A multiple-sensor discrimination system then utilizes more than one sensor to accomplish the discrimination task. In this context, the target object the system needs to select is the Re-entry Vehicle (RV), which is the lethal object in the complex. A ballistic missile launch will also yield a spent booster section, known as the tank, natural debris from the deployment of objects, and countermeasures to deceive a defense system.[7] Multiple sensors offer the ability to observe different sets of phenomena and, when properly synthesized, can improve discrimination performance.[8]

This scenario is complicated by three principal factors. First, the physics of exoatmospheric ballistic trajectories present difficult viewing conditions for each sensor that makes sensor information difficult to correlate. On-orbit objects velocities of approximately seven km/s can significantly limit the observation window of the sensors on the ground while closing velocities in excess of 10 km/s can limit the observation window for interceptors.[9] Second, Closely-Spaced Objects (CSO) present significant difficulties for sensor observation and multiple-sensor correlation as sensor measurement uncertainty makes high fidelity position determination difficult. "Closely-spaced," in this context, can be regarded as objects that are spaced closer than the measurement uncertainty band for one or multiple sensors within the observation architecture. Resolving targets from one sensor image to another can prove difficult under these conditions. Finally, the sensors, such as radar and optical sensors, are corrupted by noise and bias making discrimination still more difficult.

1.1 Problem Statement

This report describes the development of the necessary algorithms for the multiple sensor correlation and discrimination of CSOs[10,11] and the evaluation of their performance in a relevant simulation environment. These algorithms handle the sensor returns, process them into useful metrics, and assess the probability that a given object is the object of interest from these metrics. To examine the performance of these algorithms, a simulation environment was developed. It included relevant characteristics of the sensors in the discrimination system, a modeling process for the ballistic trajectories of the CSOs in the target complex, and a decision-making module containing the algorithms for handling the sensor returns and correlating and discriminating the object of interest.

The measure of discrimination performance on a ballistic missile target complex for a multi-sensor system is the probability of the ballistic missile defense system properly discriminating the RV from other objects in the target complex. This probability is known as the probability of discrimination, P_d, and depends upon the system's ability to accurately track and correlate each object in addition to correctly discriminating each object and selecting the RV. The Figure of Merit (FOM) to assess the system's correlation performance is the probability of correlation, P_c.

1.2 Nominal System Architecture

A multiple-sensor discrimination system will be comprised of at least two sensors. As dictated by the need to conduct early warning, search, sensor cueing, tracking, discrimination, and homing, a system can have additional sensors as necessary to meet these needs. However, this research is primarily concerned with analyzing the discrimination performance of the system. To this end, only the sensor components essential to discrimination will be considered. The nominal system consisted of a sea-based radar system and an optical sensor deployed in a Kill Vehicle (KV) to intercept the target. This architecture is depicted in Figure 1.

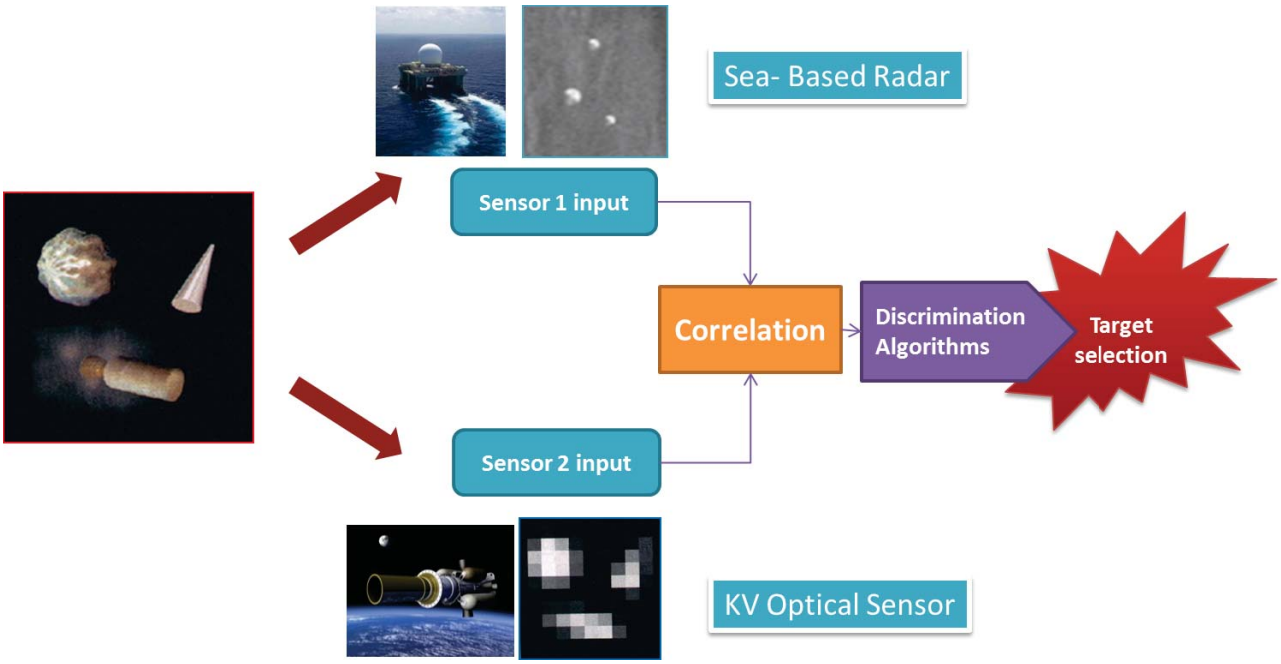

Figure 1. Nominal system architecture

The radar system was responsible for tracking the complex, providing data to calculate a launch solution to intercept the target complex, and providing discrimination information. The KV possessed an onboard optical sensor for target acquisition, tracking, and discrimination. The system fused the tracks and discrimination information from the two sensors to select a target.

2 Simulation Environment Development

Figure 2 depicts the steps performed in the simulation environment and the requisite elements for modeling that particular step. The simulation depicted the launch of a threat, followed by a radar tracking the threat, the launch of an interceptor, and finally the terminal acquisition by the Kill Vehicle (KV).

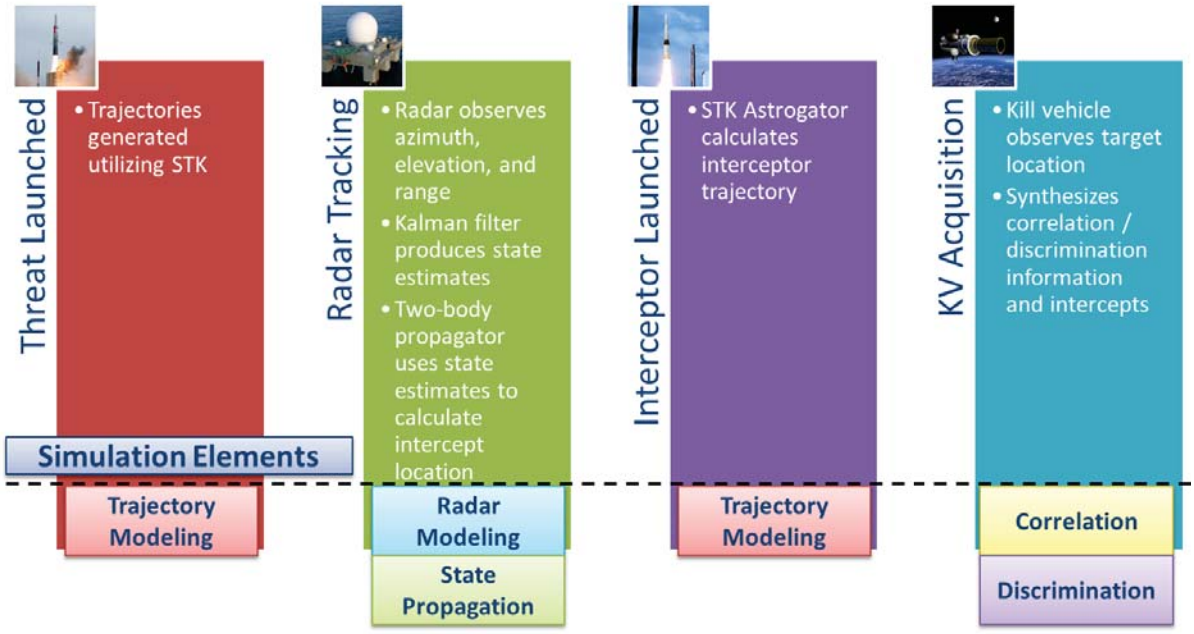

Figure 2. Simulation environment developed to support a concept of operation

2.1 Target Complex Trajectory Modeling

Two-body orbit dynamics was utilized to generate ballistic trajectories between the desired burnout and reentry points. The dispersion of object trajectories within the target complex was achieved by varying the velocity of each object at the burnout points. The generated trajectories served as the "truth" for the sensor modeling process. The System Tool Kit (STK) was utilized to calculate the trajectories of these objects. This interface is depicted in Figure 3. The interface allowed STK to be controlled iteratively and autonomously by MATLAB. Functions such as simulation control, tracking, correlation, and discrimination took place within MATLAB. The necessary information and commands were then passed to STK to generate truth and estimated trajectories and calculate the interceptor flight path. The entirety of the interface code is contained in Appendix IV. The complete interceptor trajectory, from initial launch to target intercept, was considered in the concept of operation. However, only the exoatmospheric trajectory of the target complex was modeled, starting immediately after complex deployment until re-entry.

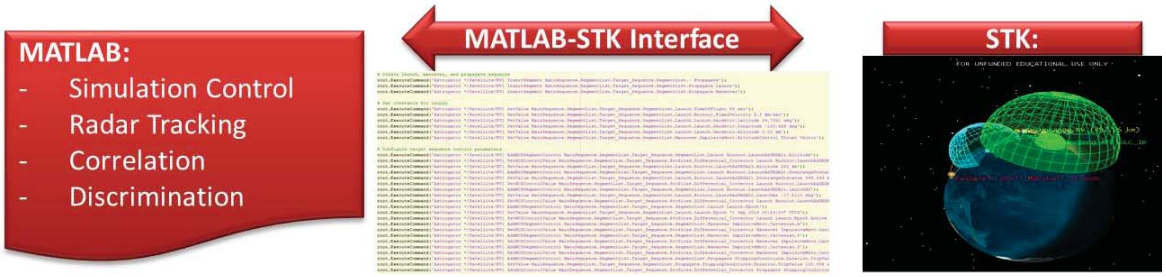

Figure 3. MATLAB – STK interface

2.2 Radar Observation Modeling

A radar system takes measurements of azimuth, elevation, and range for each object it detects, and calculates the associated rates of these measurements. To model a radar observation, measurement error was added to the truth measurements of the target complex. Measurement error consisted of two sources, accuracy and precision. Accuracy was defined as the degree to which the sensor was able to measure truth, while precision was the uncertainty associated with a measurement. Accuracy was reflected by the bias of the radar system. It was assumed that the bias was constant for the radar system, meaning the same error in azimuth, elevation, and range was added to each detected target. Precision was reflected by the random noise in each subsequent measurement. The noise was assumed to be Gaussian with a mean of zero. The standard deviation of the noise distribution was based upon the radar range and angular resolution. The measurements were then given by Equation (1), where z_k is the noisy measurement, x_k is the truth measurement, w_k is the noise at a discrete moment in time, and b_k is the bias.

$$z_k = x_k + w_k + b_k. \qquad (1)$$

The radar system's ultimate task was to track the target complex in order to estimate its ballistic trajectory. Due to the dynamic nature of the trajectory, a Kalman Filter (KF) was implemented to estimate the position and velocity of the target in the topocentric-horizon coordinate system, also known as the South, East, Zenith (SEZ) coordinate frame. The MATLAB code for the KF is contained in Appendix III. This coordinate frame was centered at the radar site. Track information was then converted into the Earth-Centered Inertial (ECI) coordinate system for orbit ephemeris determination. The SEZ and ECI coordinate frames are depicted in Figure 4.

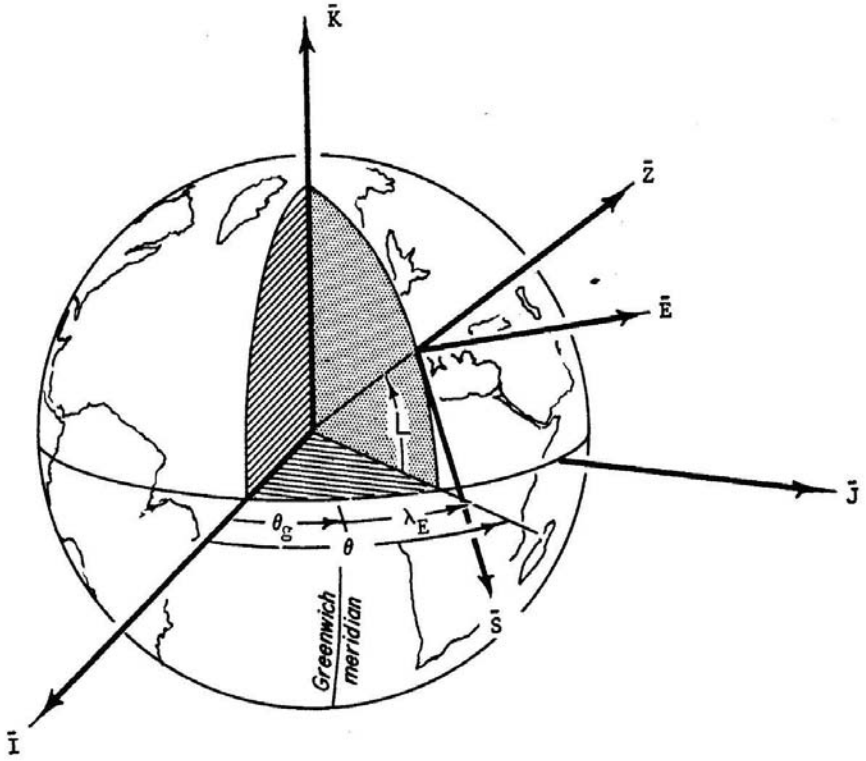

Figure 4. SEZ and ECI coordinate frames[12]

2.3 Optical Observation Modeling

Unlike radar, an optical sensor is only capable of capturing two-dimensional position information within the focal plane's field of view. In order to model an optical observation, the truth complex had to be projected onto the field of view of the KV. First, the position of the target complex was described in the KV-centered coordinate system, depicted in Figure 5, where the p-axis is along the boresight from the KV to the RV and the m- and n-axes are the perpendicular axes.

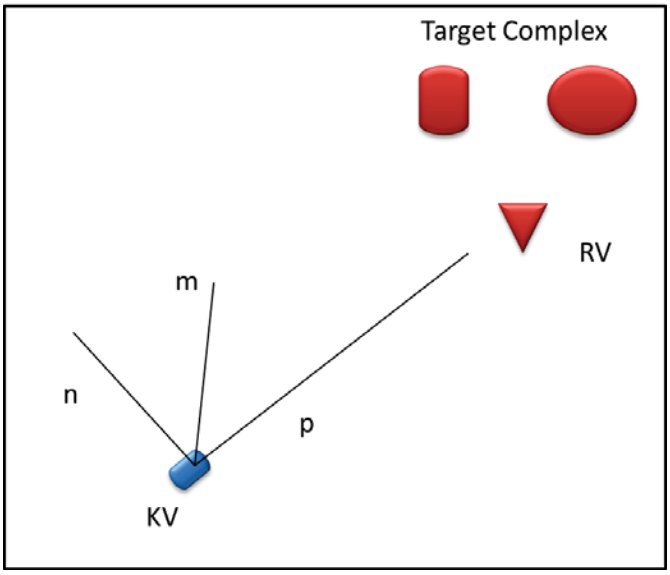

Figure 5. KV-centered coordinate system (p-m-n axes)

This description required knowledge of the truth position of the KV and the target complex at each measurement instance. The coordinate transformation from ECI to p-m-n was then accomplished using Euler angles.[13] To transform from the ECI coordinates into the p-m-n frame, the relative position of the target complex from the KV was calculated in ECI, using Equation (2). $\underline{P}$ is the relative position of the target complex from the KV, $\underline{R_t}$ is the absolute position of the target complex, and $\underline{R_i}$ is the absolute position of the KV. A similar process was utilized to calculate the relative velocity of the target complex.

$$\underline{P} = \underline{R_t} - \underline{R_i} \tag{2}$$

With the relative position and velocity known, the ECI coordinates could then be transformed into p-m-n using a Direction Cosine Matrix (DCM), which represents rotation about the 3 (K) axis followed by rotation about the 2 (J) axis. First, the ECI frame was rotated about the **K**-axis by α, given by Equation (3), where *a* and *b* are the **I** and **J** components of the relative position, respectively.

$$\alpha = \tan^{-1}\left(\frac{b}{a}\right) \tag{3}$$

Next, the coordinate frame was rotated around the **J**-axis by β, given by Equation (4), where *c* is the **K** component of the relative position, and *d* is the magnitude of the **I-J** components.

$$\beta = \tan^{-1}\left(\frac{-c}{d}\right) \tag{4}$$

The DCM was then assembled from the constituent components, as shown in Equations (5) - (7), where **D** is the DCM used to rotate from ECI to p-m-n.

$$\boldsymbol{D_3} = \begin{bmatrix} \cos \propto & \sin \propto & 0 \\ -\sin \propto & \cos \propto & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$\boldsymbol{D_2} = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \quad (6)$$

$$\boldsymbol{D} = \boldsymbol{D_2 D_3} \quad (7)$$

With the DCM known, the p-m-n position could be calculated via Equation (8), where $\underline{P'}$ is the position of the target complex in p-m-n coordinates. Velocity could be found in a similar manner.

$$\underline{P'} = \boldsymbol{D} * \underline{P} \quad (8)$$

With three-dimensional position and velocity known, range, azimuth, and elevation were calculated based upon the Cartesian p-m-n elements. The optical observations were assumed to be the truth observations.

3 Correlation and Discrimination Algorithms

3.1 Target Object Map Generation and Correlation

With the state estimates determined by the radar and subsequently observed by the KV optical sensor, the information from these two sources needed to be correlated. To accomplish this, the system must be able to associate the different sensor returns from the same object. A method known as anchor Target Object Map (TOM) is used. A TOM is essentially a picture of the target complex at a discrete time generated by a sensor. The TOM is a matrix containing the coordinates of the location of the center of mass of each of the detected objects in the target complex in a reference coordinate frame. TOMs from different sensors can then be compared provided that each TOM reflects similar viewing geometry and the same instance in time. Because the sensors within the architecture rarely observe a target at the same instance from the same location, a method was developed to generate similar TOMs both temporally and geometrically. The MATLAB code utilized for this method is contained in Appendix V.

The radar TOM was generated using the tracks from the radar observation. First, the target complex track was projected from the time of minimum track uncertainty, defined as the normal of the variance elements of the track in the KF estimates, to the desired time. With the anticipated position known, the coordinates of the target complex were then rotated and projected into the focal plane of the KV, similar to the procedure described in Section 2.3. The optical TOM could be obtained directly from the KV observations since the radar TOM is projected to the orientation and time during which the KV is observing the complex.

With a TOM generated for each sensor, the TOMs had to be correlated as depicted in Figure 6. First, the sensor bias was eliminated by "anchoring" each TOM to a reference object. The reference object is one readily identifiable by both sensors. The anchor was selected using the probabilistic discrimination algorithm described in Section 3.2. Each sensor independently performed discrimination to select its own anchor for its TOM. After selecting an anchor, the anchor was set as the origin of the TOM by subtracting its coordinates from the coordinates of all the objects.

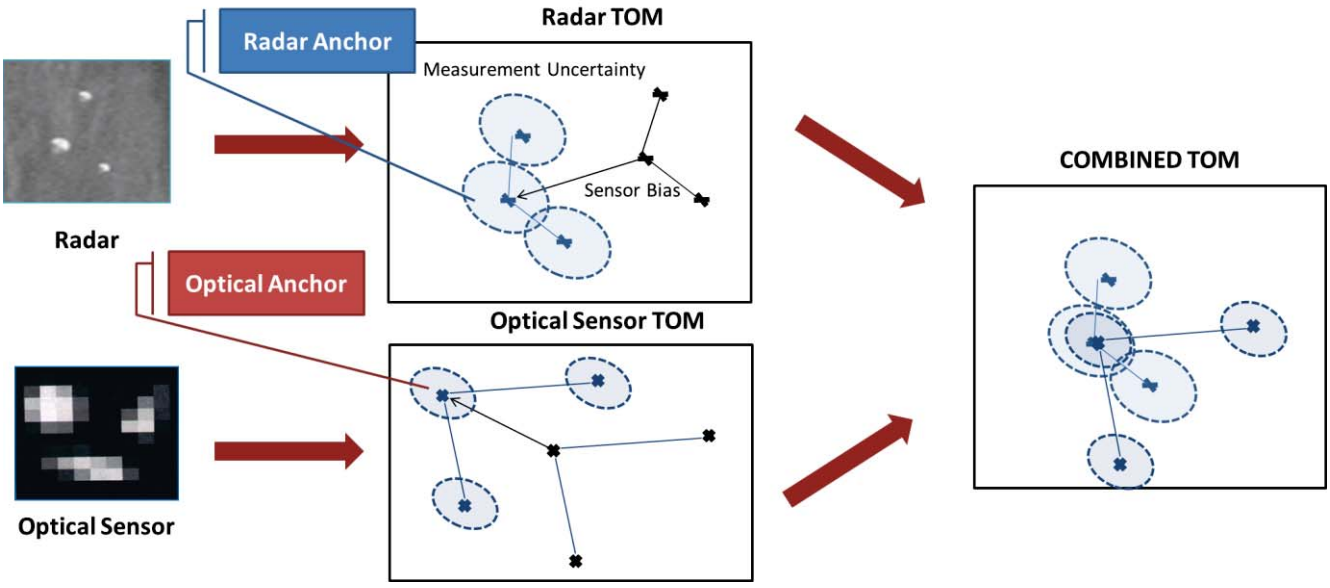

Figure 6. TOM anchoring and correlation processes

The two TOMs were then superimposed. Due to sensor uncertainty, the observed location of each object may differ between the two sensors. As the spacing between the objects decreases, the relative arrangement of the target complex may vary between two TOMs. The differing in target position makes the association of objects difficult. To correctly associate the targets, each possible configuration of associations was tested. An example of this correlation process for four active tracks from the radar and the optical sensor is depicted in Figure 7. Pairings between objects are depicted by the green lines between objects.

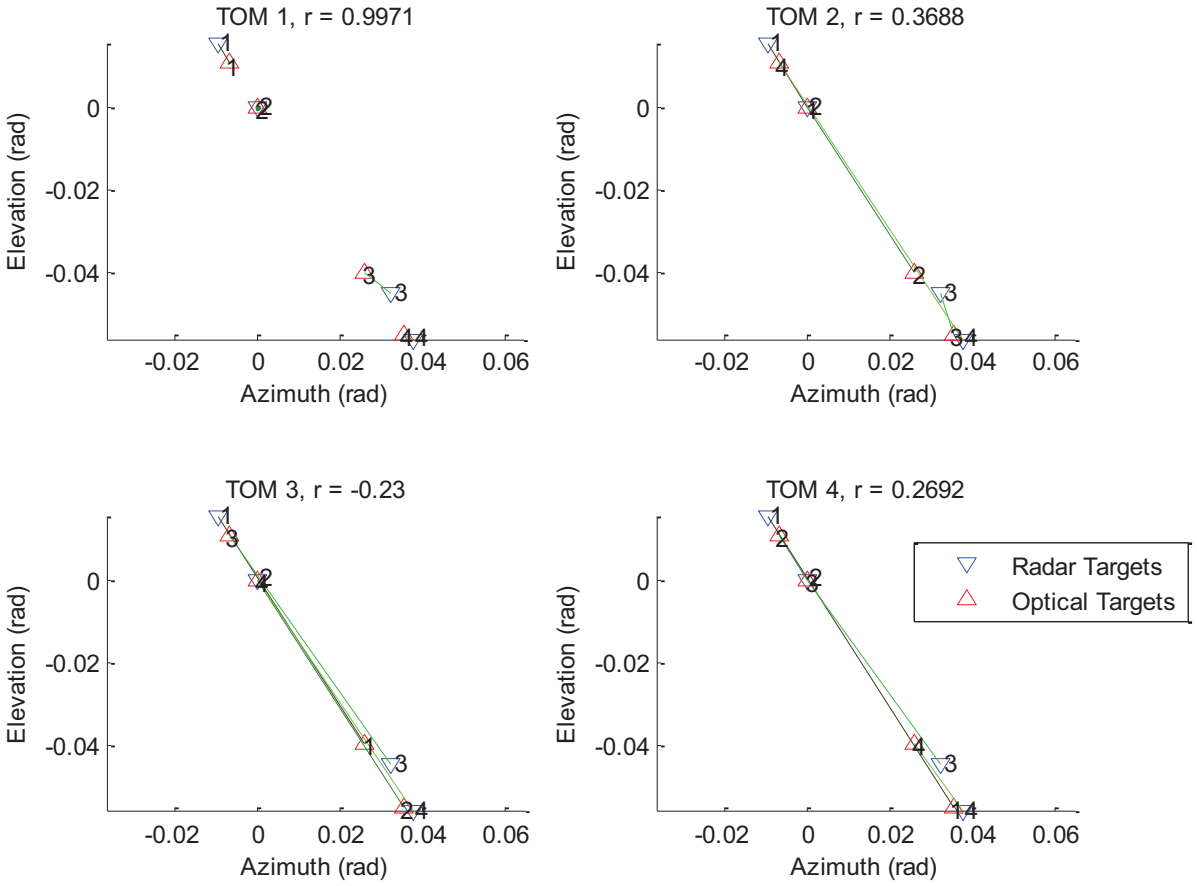

Figure 7. Possible track correlation pairings for four objects

For each arrangement, a correlation coefficient, r, was calculated using Equation (9), where A and B are the matrices containing the coordinates within the TOM and m and n are the number of observations within A and B, respectively. Maximizing r corresponded to minimizing the Euclidean distance between the pairings in the TOM. A nominal TOM A matrix is depicted in Table 1. Note that the track numbers (shaded in red) are only for reference and are not included for the purposes of calculating r. In this case, track 2 was identified as the anchor and as such is set as the origin (0, 0). $\bar{A}$ and $\bar{B}$ are the mean values of the A and B matrices.

Table 1. Example TOM A matrix

Track	Azimuth (rad)	Elevation (rad)
1	-0.014	0.015
2	0	0
3	0.030	-0.045
4	0.039	-0.049

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (9)$$

The track correlation pairing generating the greatest *r* was then selected and the respective tracks are associated.

3.2 Discrimination Algorithm

The discrimination algorithm was responsible for selecting a target from the sensors' discrimination information. The algorithm used Dempster-Shafer decision logic, which is explained in Section 3.2.1 – Section 3.2.4. The MATLAB implementation of this algorithm is contained in Appendix VI.

3.2.1 Dempster-Shafer Decision Logic

As previously stated, discrimination is the process of transforming measurements into a decision. In the context of this project, the discrimination algorithm is utilized to select the RV given a set of measurements. Dempster-Shafer logic was employed within this algorithm, as the Dempster-Shafer algorithm "is useful when the sensors contributing information cannot associate a 100 percent probability of certainty to their output decisions."[14] The Dempster-Shafer method captures and combines the certainty each sensor possess in its object discrimination capabilities.[15] Uncertainty in sensor measurement is of particular concern when discriminating targets on a ballistic trajectory. The extreme geometry of the problem, the associated temporal constraints, and ambiguous and imperfect sources of information all contribute to the uncertainty in selecting the RV from the target complex.

Dempster-Shafer involves the combination of data from a set of sensors. Each sensor observes a certain set of characteristics and generates returns from these observations. From these observations, the probability of each measurement resulting from a certain type of object, or object class, is calculated. This probability of each class is used to calculate a belief in a given hypothesis, known as an Evidence Measure (EM) that reflects the relative certainty of selecting a certain hypothesis. The hypotheses are broken into two categories: singleton and combination. Singleton hypotheses categorize the observation into one object, such as an RV. Combination hypotheses categorize the observation into multiple objects, e.g. RV or decoy. An EM of unity indicates absolute certainty in a certain hypothesis. The EMs from every sensor are then combined utilizing Dempster's rule of combination. The hypothesis with the greatest amount of accumulated EM is then selected.

3.2.2 Calculating Evidence Measures

In order to calculate the EM for each observation, the distribution of measurements must be known. This project assumes this distribution to be Gaussian, with the expected value for a given class to be the mean of the distribution, μ.[16] The sensor measurement uncertainty determines the standard deviation of the distribution, σ. From the Gaussian distribution, the probability of a given measurement stemming from a given object can be calculated. Figure 8

depicts an observation and the calculation of the probability of measurement for a three-class hypothesis example. This observed characteristic can be anything the sensor is capable of measuring, such as the size or a dynamic characteristic of the object. This project assumed the sensors are equally capable of measuring any class of object. As such, the characteristic distributions are symmetric but possess different expected values. Table 2 shows the calculated probabilities from these distributions.

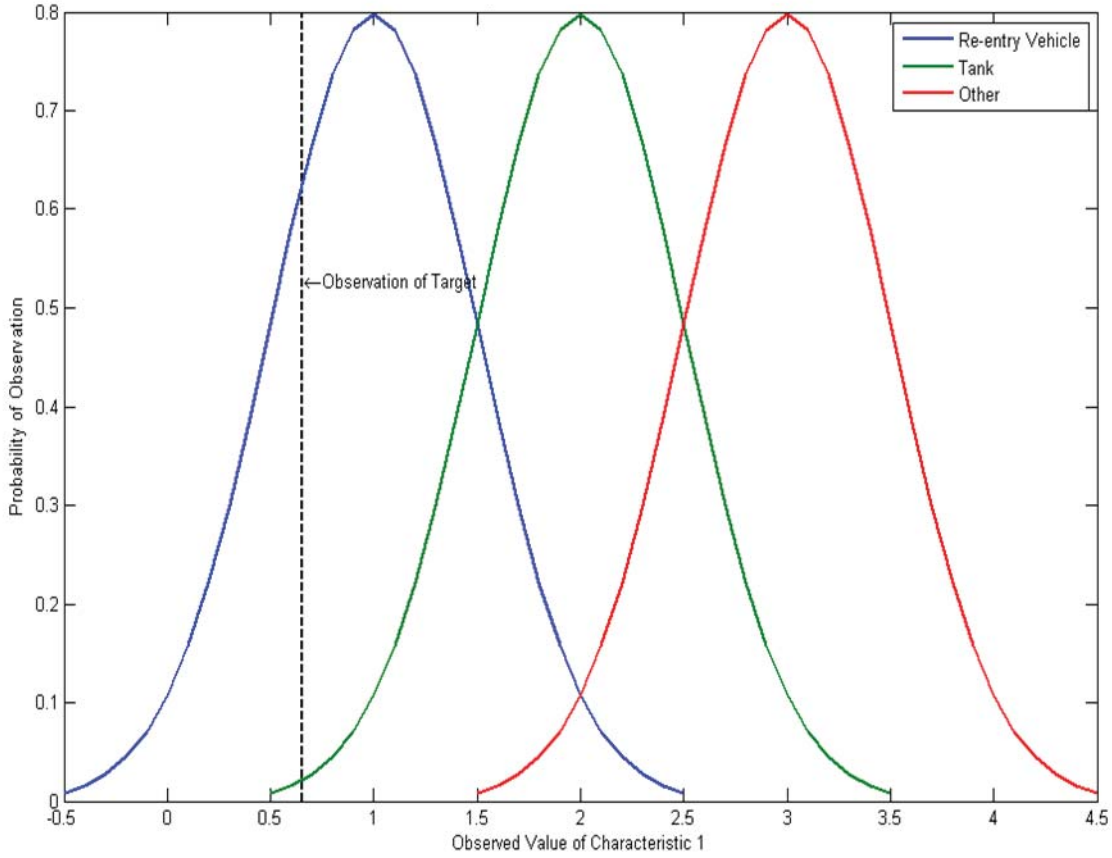

Figure 8. Example calculation of measurement probabilities for three-class hypotheses

Table 2. Example calculated probabilities

P(X|RV)	0.629
P(X|Tank)	0.021
P(X|Other)	~0

A given set of EMs must then be assigned to support a series of hypotheses. The set of all the possible hypotheses is referred to as the frame of discernment, denoted by Θ. Any EM can be assigned to any of the propositions, $a_1, a_2, \ldots, a_n$, or any of the possible unions, e.g. $a_1 \cup a_2$, or finally to Θ itself. A given EM can only be assigned to the extent of the confidence in the

measure. For example, if a sensor is 80 percent confident a measurement indicates a_1, an EM of 0.8 is assigned to a_1. The remainder, 0.2, is assigned to Θ, which represents all the possible propositions simultaneously, or uncertainty.

3.2.3 Evidence Measure Combination

With the proper EM assigned to each hypothesis for each sensor observation, the EMs can be combined to generate the total support for each hypothesis using Dempster's rule. Dempster's rule is implemented by forming a matrix with the elements to be combined in the first column and first row. The combinations to be formed are those hypotheses which have an intersection that exists. For example, hypotheses $a_1 \cup a_3$ and $a_3 \cup a_4$ intersection is a_3. Combining the EMs for these two hypotheses would yield evidence for a_3.[17] The matrix elements and subsequent combinations are made by multiplying the column element in the first column with the row element in the first row. An example for two sensors, sensor A and sensor B, is shown in Table 3. For additional sensors, the above method is repeated for each subsequent sensor.

Table 3. Application of Dempster's Rule[18]

	$m_B(a_3 \cup a_4) = 0.7$	$m_B(\Theta) = 0.3$
$m_A(\Theta) = 0.4$	$m(a_3 \cup a_4) = 0.28$	$m(\Theta) = 0.12$
$m_A(a_1 \cup a_3) = 0.6$	$m(a_3) = 0.42$	$m(a_1 \cup a_3) = 0.18$

Sensor A Evidence Measure
Sensor B Evidence Measure
Combined Evidence Measure

A special case of combination occurs when measures support mutually exclusive hypotheses, for example a_1 and $a_2 \cup a_3$. In this case, the product of these two EMs is assigned to the empty set, ϕ, which represents measures that support different and exclusive hypotheses. ϕ, also known as conflict, is the complement of Θ, which represents a measure that could represent any and all hypotheses. The empty set ϕ is only pertinent in the combination of EMs, as a single measure can be uncertain in what hypothesis it supports but cannot conflict itself.

3.2.4 Resolving Conflict and Uncertainty

While knowledge of a degree of conflict and uncertainty in some applications may prove useful, they are not consistent with BMD discrimination. In this context, uncertainty and conflict must be eliminated and assigned to support a hypothesis. To do so, one must calculate the "pignistic probability" of a certain hypothesis from the given EMs. Pignistic is derived from the Latin term *pignus*, meaning a bet, and is indicative of the nature of Pignistic Probabilities (PP).[19] PPs are not analytical probabilities in the sense that a PP of 0.5 would not indicate an event is expected to occur 50 times in 100 trials. Rather, they are meant to guide a "bet," or decision, to a particular hypothesis. Their absolute value is irrelevant, only their relative magnitudes matter. For example, consider the two PPs for hypotheses a and b, where $PP_a = 0.75$ and $PP_b = 0.25$. The pignistic probability of a does not indicate three times greater confidence in a, merely that a should be selected over b.

To calculate the PP for a given hypothesis, conflict is first addressed. To eliminate conflict, the EMs for the non-empty set hypotheses are normalized by a factor, K, given by

Equation (10).[20] K is a measure of the total amount of conflict between two measurements, where a K of unity indicates no conflict and an undefined K indicates complete conflict between the measurements, causing the sum given by Dempster's rule to be undefined.

$$K^{-1} = 1 - \sum_{a_i \cap b_j = \emptyset} \left[m_a(a_i) m_b(b_j) \right] \qquad (10)$$

After normalizing, the conflict terms are set to zero. With conflict eliminated, the EMs for combination hypotheses and uncertainty are then equally divided between their constituent components. For example, if $m(a_1 U a_2) = 0.5$, $m(a_1)$ and $m(a_2)$ would both be assigned 0.25. The uncertainty EM would then be divided equally among all the hypotheses. At this point, the remaining hypotheses are the singleton elements (a_1, a_2, …). The evidence measure now assigned to these elements is the PP for each hypothesis. With conflict, uncertainty, and the combination hypotheses eliminated, the hypothesis with the highest total PP is then selected.

4 Nominal Scenario Simulation and Results

In order to support future decisions regarding investment in and improvement of the ballistic missile defense system, the effect of varying system parameters on P_d was assessed using a simulation environment developed as a part of this project. The simulation environment depicted a launch event of a threat, tracking by a radar system, and finally engagement by an interceptor. The simulation then assessed the success or failure of the engagement. The nominal scenario and one iteration of the simulation are described in Section 4.1 through Section 4.4.

4.1 Nominal Simulation Scenario

To assess the created simulation environment, a nominal scenario was developed that was indicative of a possible engagement of a closely-spaced target complex. The scenario involved the launch of a ballistic missile from the Reagan Test Site in the Kwajalein Atoll with a destination of Seattle, WA. The missile was assumed to deploy the target complex at an altitude of 100 km directly above the Reagan Test Site and re-enter the atmosphere at an altitude of 100 km directly above Seattle. The target complex consisted of a RV, a tank, and two decoys. The trajectories of the complex objects are shown in Figure 9, with the RV, tank, and decoys trajectories shown in red, orange, and yellow, respectively.

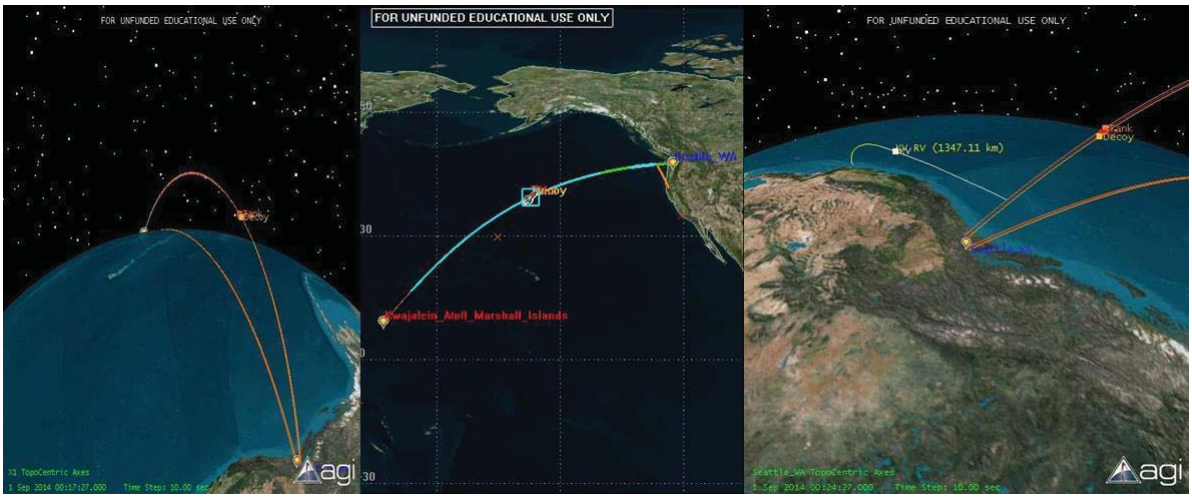

Figure 9. Nominal scenario ballistic trajectories[21]

The defensive system consisted of a sea-based X-band radar (X1) north of Hawaii and an interceptor launched from Vandenberg AFB, CA, that deployed the KV. The interceptor was assumed to have a range of 2500 km, while X1 was assumed to have a range of 4500 km. The interceptor attempted to engage the target at an altitude of 250 km. The KV was assumed to acquire the target at a range of 500 km and tracked the complex as it closed. The radar and interceptor coverage are depicted in Figure 10, with X1 coverage shown in light blue and the interceptor shown in light green. In addition, the portion of the target complex trajectory that can

be tracked by X1 is highlighted in light blue and the portion that is in interceptor range highlighted in light green.

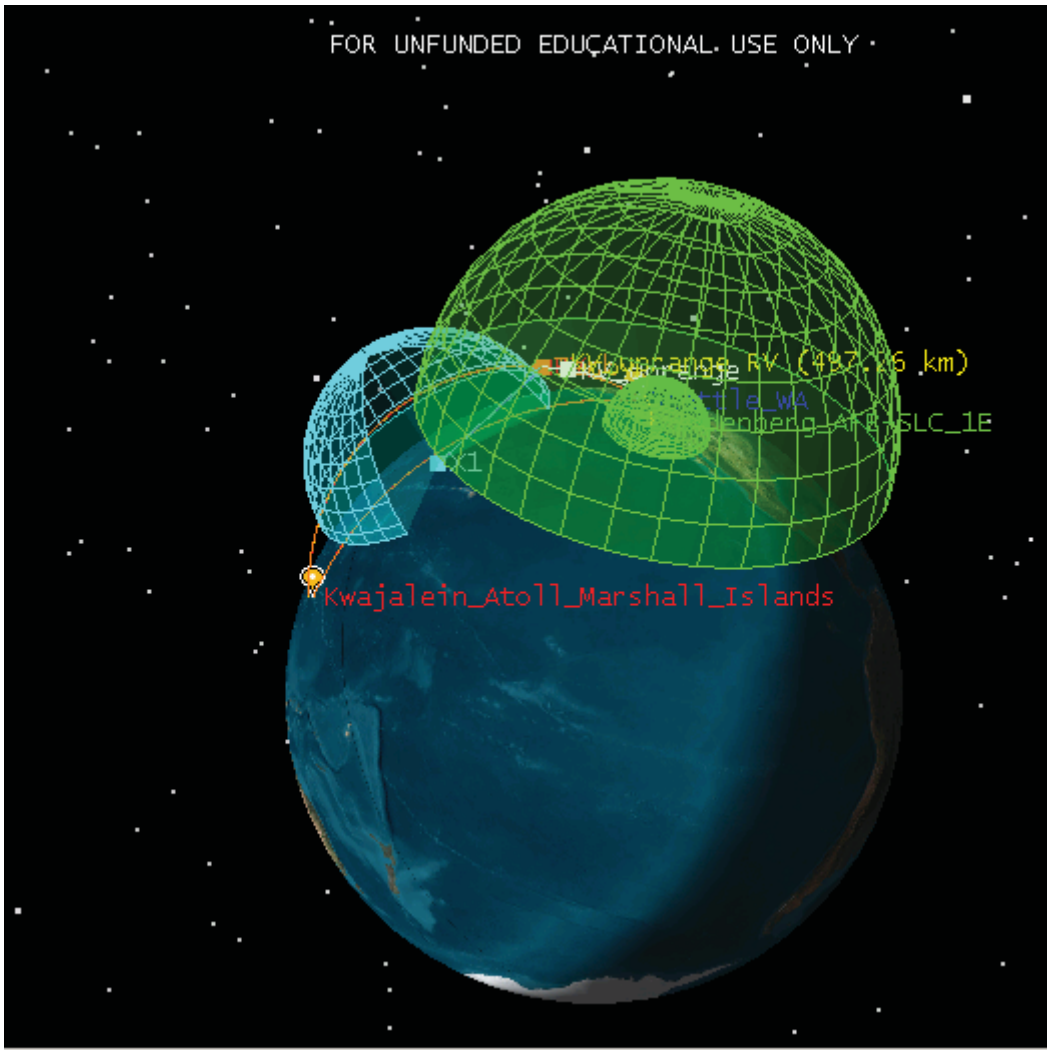

Figure 10. X1 and interceptor coverage

Using the modeling techniques described in Section 1.2, an engagement of the target complex was simulated. The defensive system was responsible for tracking the complex with both the radar (X1) and the KV optical sensor, correlating the tracks, fusing the discrimination information, and ultimately selecting a target.

The following describes the results of the simulation using the nominal scenario described above.

4.2 Radar Kalman Filter and Tracking Performance

As mentioned in Section 2.2, a Kalman filter was employed in order to estimate the target complex states from the noisy radar measurements. The performance of the filter was measured by two metrics, the residuals of the estimates compared to the truth measurements and the

variance of the state estimates. Figure 11 illustrates the position and velocity residuals of the estimates for the RV track versus the time after launch.

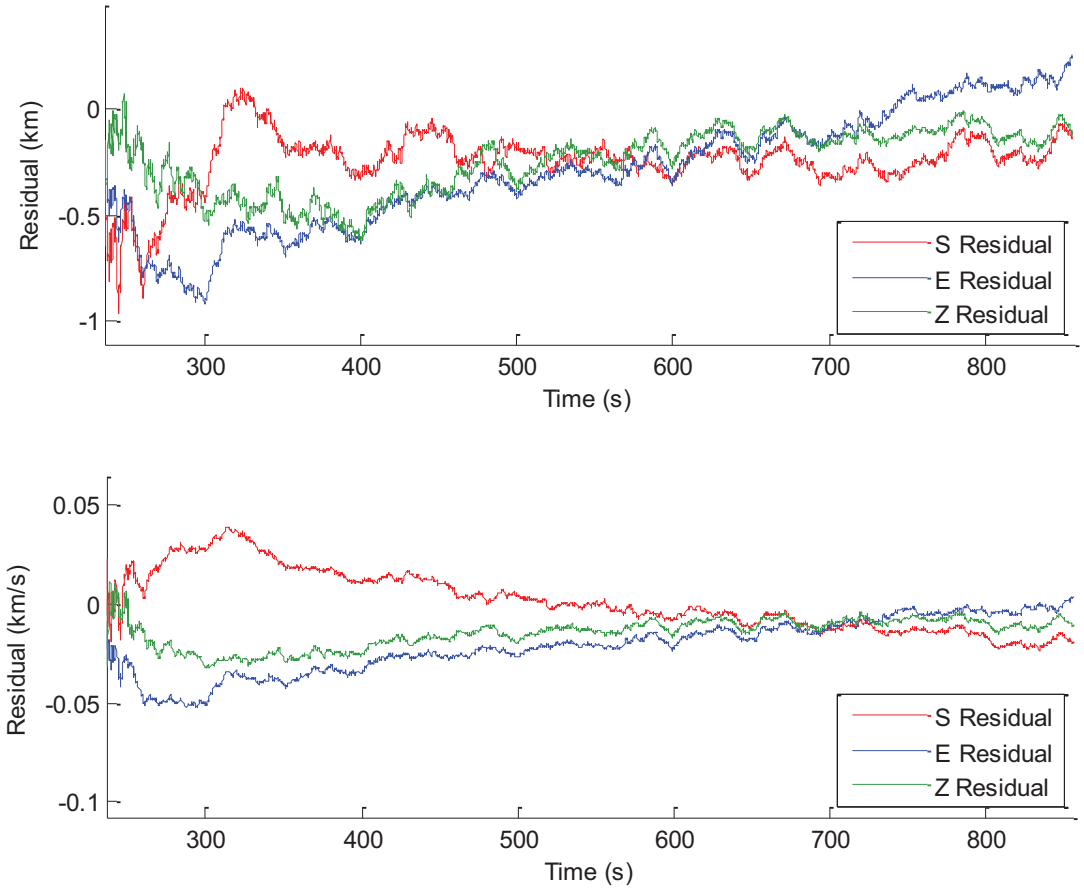

Figure 11. Residuals of position and velocity estimates for the RV track

After an initialization period (not shown), the position residual gradually decreased to within a magnitude approximately 0.5 km. The velocity residuals fall to within 0.03 km/s. Both the position and velocity residuals improve as the target nears 700 seconds after launch. This is a result of the range to the target decreasing which decreases the covariance of the measurements. The residuals then diverge after the target passes through the closest point of approach as the range begins to increase. This is of particular concern for correlation purposes as larger residuals and the associated larger errors make associating sensor tracks more difficult since the anticipated position of an object and its actual position will vary by a larger amount. After approximately 700 seconds after launch, the range to the target begins to increase. This drives an increase in the residuals.

Figure 12 depicts the variance of the position estimates for track 3. The variance of the estimates exhibited the same behavior as the residuals, decreasing in magnitude as the target range decreased and growing as the range to the target increased.

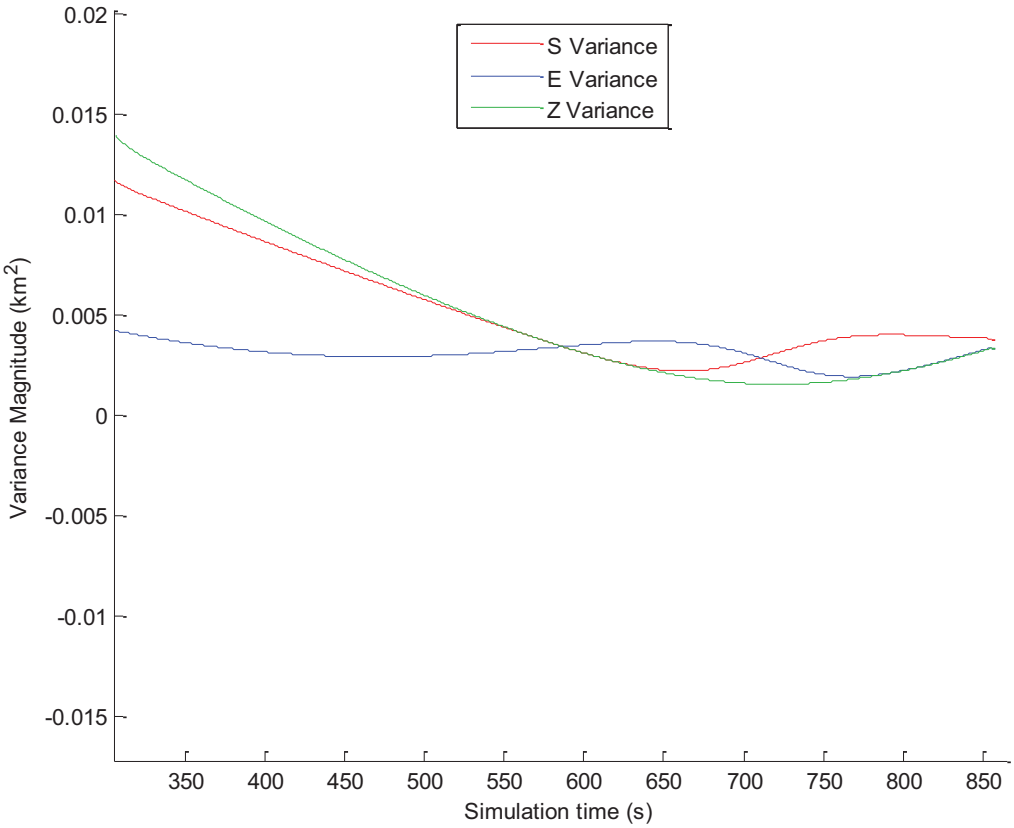

Figure 12. Variance of SEZ position estimates

4.3 TOM Correlation

Since the optical measurements were assumed to be truth, tracking and filtering was unnecessary. The KV TOM was obtained directly at the correlation time from the optical observations. The estimated radar states at this time were then rotated and projected onto the focal plane of the KV. First, an anchor was selected by both sensors based upon their discrimination information. The anchor to be identified was the tank, as it is typically the largest object in the target complex and most easily identified by both sensors. In this example, both sensors properly selected the tank as the anchor. This allowed the TOMs to be properly anchored. The tracks then were correlated. Figure 7 in Section 3.1 depicts the possible pairings. Configuration one yielded an r value of 0.997, while configuration two, three, and four yielded 0.369, -0.23, and 0.269, respectively. As such, configuration one was selected. This turned out to be the correct correlation.

4.4 Discrimination

With the tracks from the two sensors correlated, discrimination was then performed. Each sensor observed one discriminable characteristic. The probability distributions for the radar and

optical characteristics are shown in Figure 13 and Figure 14, respectively. The shape of these distributions is based upon the expected value for the given characteristic for each class and the standard deviation of the measurements from the sensors.

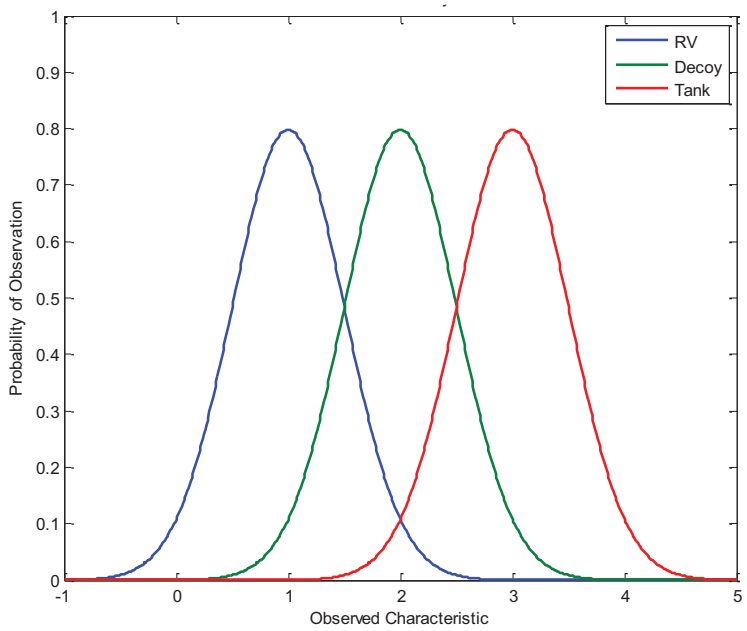

Figure 13. Radar characteristic probability distributions for three classes

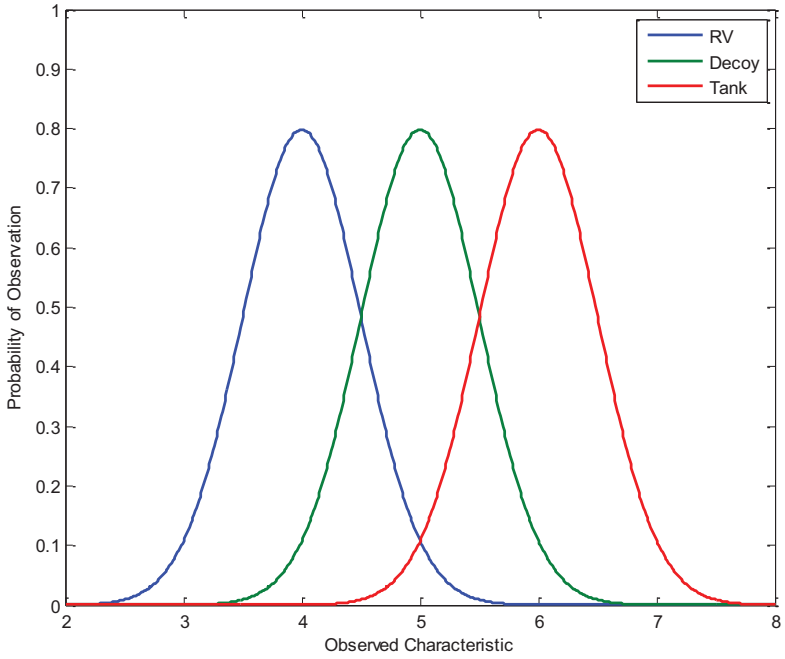

Figure 14. Optical characteristic probability distributions for three classes

The radar and optical discrimination characteristic observations are contained in Table 4.

Table 4. Example scenario radar and optical observed discrimination characteristics

Track	Radar Characteristic	Optical Characteristic
1	1.308	4.094
2	3.234	5.748
3	1.634	5.357
4	2.546	4.625

From these observed characteristics, the probability of observation for each class was calculated for each set of measurements and is contained in Table 5 and Table 6, respectively.

Table 5. Example scenario probability of observation from radar characteristic observation

Track	Probability of RV	Probability of Tank	Probability of Decoy
1	0.660	0.003	0.306
2	3.69e-5	0.715	0.038
3	0.357	0.019	0.611
4	0.007	0.529	0.440

Table 6. Example scenario probability of observation from optical characteristic observation

Track	Probability of RV	Probability of Tank	Probability of Decoy
1	0.784	~0	0.155
2	0.002	0.703	0.260
3	0.020	0.350	0.618
4	0.365	0.018	0.603

Based upon these distributions and the measurements collected by the two sensors, EMs were calculated and combined with Dempster's rule of combination. The resulting PPs are contained in Table 7. Target 1 was identified as having the greatest PP for RV and was selected as the target. In this simulation, target 1 corresponded to the RV. In selecting target 1, the system properly discriminated the target complex and selected the proper object for engagement.

Table 7. Resulting pignistic probabilities for engagement

Target	PP(RV)	PP(Tank)	PP(Decoy)
1	0.868	0.025	0.107
2	0.028	0.896	0.076
3	0.123	0.122	0.755
4	0.149	0.606	0.193

The engagement as a whole was successful because the system successfully tracked with both sensors, correlated the tracks, and properly discriminated the complex with the available

information. While this engagement was successful, the system could still fail when observing the same objects and produce an incorrect discrimination decision. This could stem from the sensor noise corrupting the TOMs or random variation in the observed characteristics.

5 Sensitivity of Sensor Resolution and Characteristic Overlap on Discrimination

This project further assessed the effect of varying radar angular and range resolution, discrimination characteristic overlap (the amount of shared area between the probability of observation distributions for the different classes), target complex size, and number of target complex objects on the performance of the defense system. The developed simulation environment was utilized to perform a Monte Carlo simulation, varying the parameters listed above. The MATLAB code utilized to run the simulation is contained in Appendix I and II. The ultimate measure of success was P_d. Other metrics were utilized to reflect the performance of the tracking, correlation, and discrimination elements of the system as necessary to provide a more nuanced perspective on the performance of the system.

5.1 Effect of Radar Resolution

The accuracy of the radar measurements in azimuth, elevation, and range can affect the radar's ability to produce quality tracks for the engagement of the threat by the interceptor. In order to assess this effect, the radar range and angular resolution were varied by +/- 10 percent of the baseline values. Elevation resolution was held at 2.3 times azimuth resolution as angular resolution varied. Range and angular bias were set with a uniformly distributed variation with a maximum magnitude equal to the range and angular resolution, respectively. Twenty-five different combinations of values were tested with three hundred individual trials conducted for each individual combination of values. The baseline parameter values for the X-band radar are contained in Table 8.

Table 8. Baseline radar parameters

Parameter	Value
Pulse repetition frequency (Hz)	45
Azimuth resolution (deg)	0.44
Elevation resolution (deg)	1.01
Range resolution (m)	10

Figure 15 depicts the sensitivity of probability of discrimination with varying range and angular resolution. The observed probabilities ranged from 0.9 to 0.95. However, the opposite of the anticipated trend was noted, as P_d actually increased as range and angular resolution degraded. The difference between minimum and maximum P_d varied by approximately five percent, a relatively small variation that may be random rather than reflective of system performance. In addition, the implementation of the Kalman filter and the correlation process may mitigate the effects of degrading sensor performance. The Kalman filter computed a weighted average between the projected position of a track and the observed position. By utilizing a projected state, the filter was able to reduce the uncertainty in the measurements to less than the uncertainty associated with a direct measurement. The correlation process proved able to successfully correlate even imperfect TOMs. Finally, incorrect correlations between RVs

and decoys may have reduced the impact of associating the wrong tracks. Since the discrimination characteristics for the RV and decoys are relatively similar, a correlation between an RV and a decoy may still produce a correct discrimination decision. Since the complex contained two decoys, one RV, and one tank, it is more likely in the event of a failed correlation that an RV and decoy track be paired than an RV and tank.

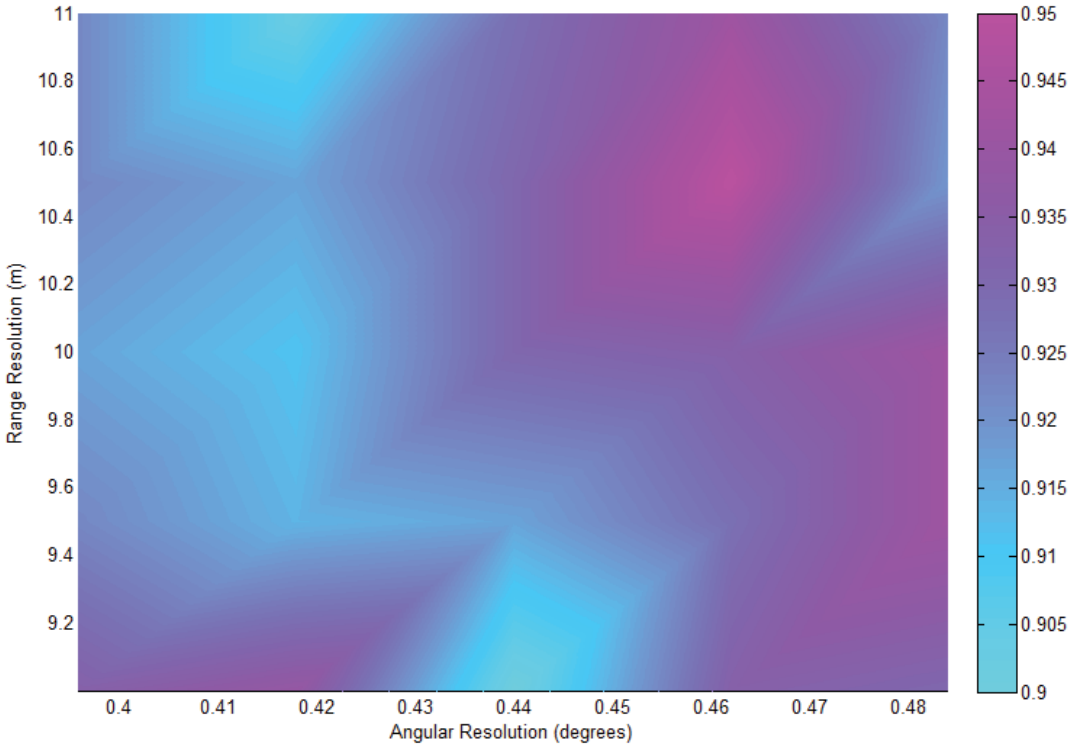

Figure 15. Variation of probability of discrimination with radar range and angular resolution

Figure 16 depicts the variation of probability of correlation with radar range and angular resolution. The observed P_c ranged from 0.94 to 0.985. With small exceptions, P_c increased as angular and range resolution decreased. This follows expectations, as improving resolution should improve the ability of the radar to produce higher quality tracks that will more closely match the scene observed by the KV upon target acquisition.

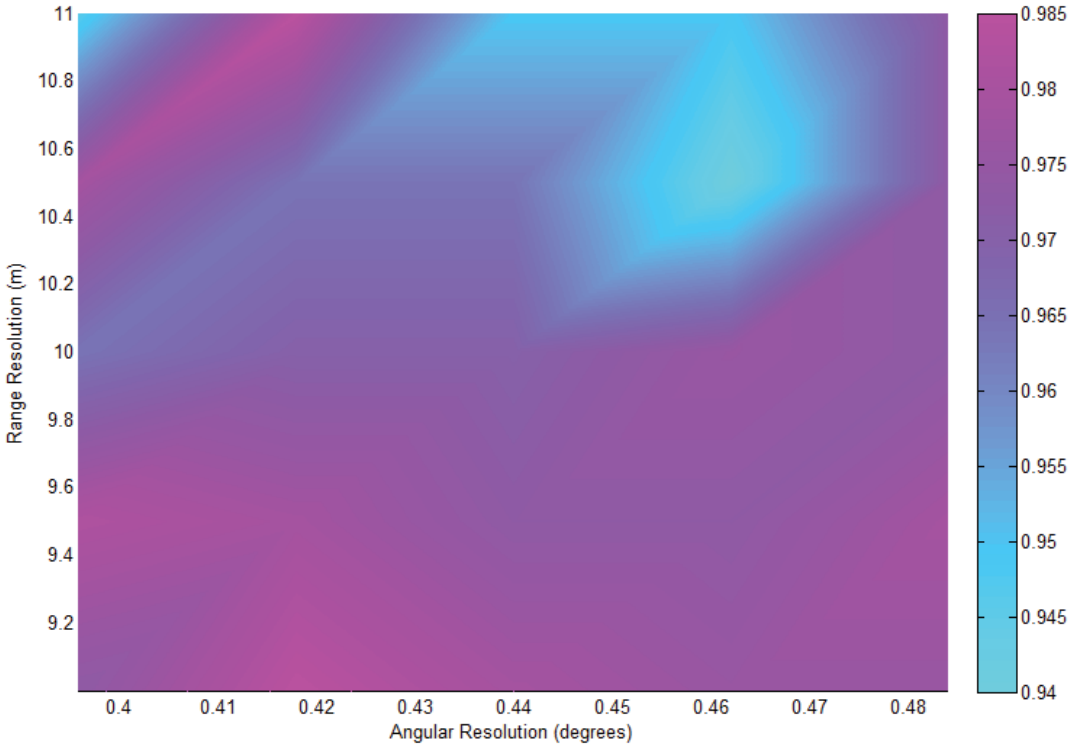

Figure 16. Variation of probability of correlation with radar range and angular resolution

Figure 17 depicts the variation of mean track uncertainty at propagation with radar resolution. Greater track uncertainty at propagation can lead to states propagated forward to the intercept time with greater uncertainty and errors, leading to a more difficult correlation process. Degrading angular resolution resulted in greater track uncertainty. This is expected, as less precise sensors produce less certain measurements. Angular resolution had a much larger effect on track uncertainty than did range resolution. Due to the range of observation (~2500 km), a small change in angular resolution produced a much larger effect on the position uncertainty. For example, to produce the same magnitude in Cartesian position uncertainty as a 0.1 degree degradation in azimuth resolution at a range of 2,500 km, range resolution would need to degrade by 4,360 m.

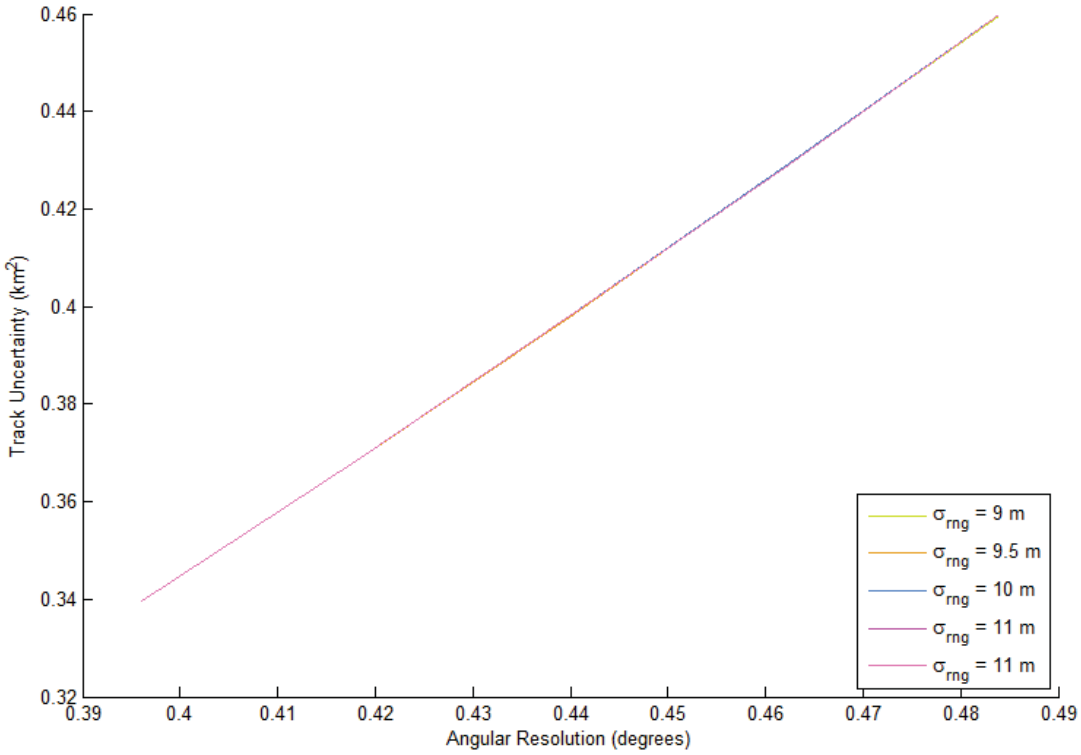

Figure 17. Variation of mean track uncertainty at propagation with radar resolution

Figure 19 depicts the variation in Zero Effort Miss distance (ZEM) with radar range and angular resolution. ZEM is defined as the Pythagorean distance between the aim point of the KV and the actual location of the selected target at intercept and is depicted in Figure 18.

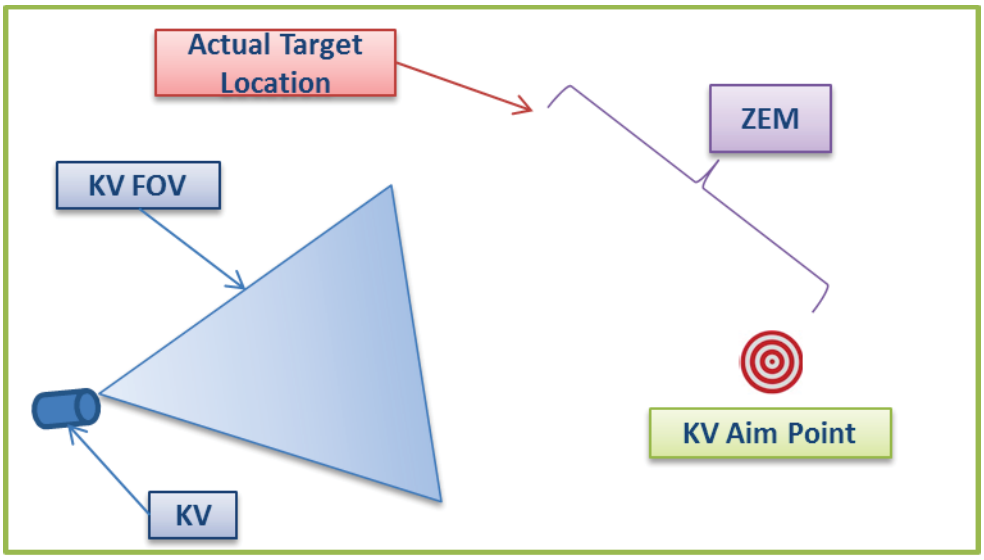

Figure 18. Zero Effort Miss distance and KV field-of-view

Greater ZEM requires a more maneuverable KV to reach the target and a greater on-board sensor FOV to detect the target. As angular resolution increased, the ZEM increased. Again, as shown in Figure 19, the effect of angular resolution was shown to be much greater than range resolution. This trend in ZEM indicates that a more accurate radar will place smaller demands on the KV as it acquires and homes on the target.

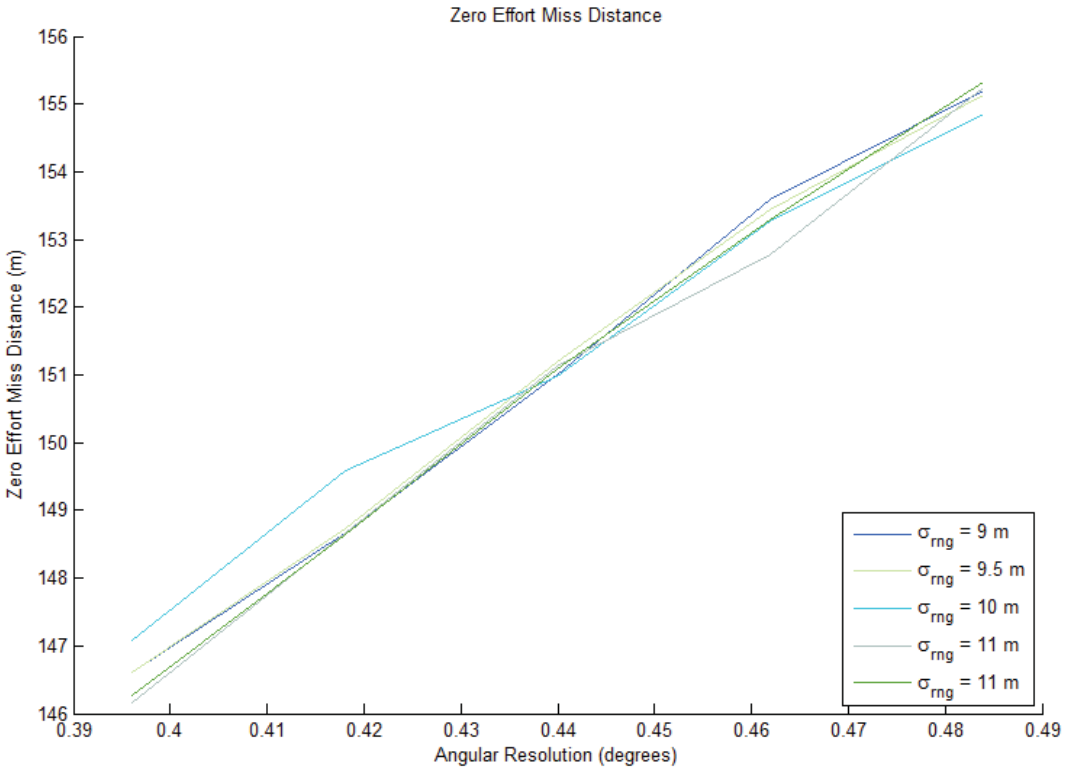

Figure 19. Variation of zero effort miss distance with radar range and angular resolution

In summary, improving radar range and angular resolution improved the ability of the system to reduce the uncertainty in the radar tracks, properly correlate the radar and KV TOMs, and reduce the demands on KV at intercept. However, improved radar resolution did not translate into a noted improved ability to discriminate the RV. This may indicate the robust ability of the Kalman filter and the correlation algorithm to mitigate the effects of degrading sensor performance. Removing the Kalman filter from the simulation and propagating the observed states from the radar would allow for direct observation of the effect of range and angular resolution on system performance.

5.2 Discrimination Characteristic Overlap Area

The amount of overlap between the probability of observation distributions for different classes is indicative of the likelihood that a given observation could indicate different classes. An example of characteristic overlap is shown in Figure 20.

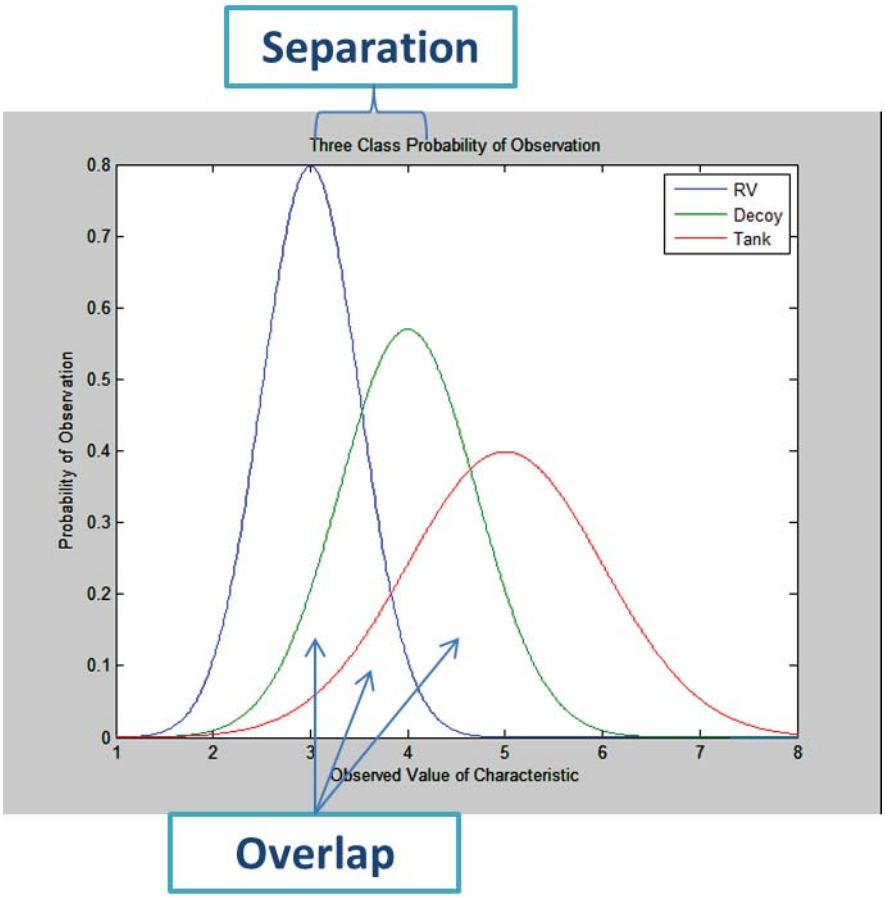

Figure 20. Example of discrimination characteristic overlap area and separation

The distributions tend to move together for two reasons. First, less capable sensors result in distributions with higher standard deviations. Greater standard deviations tend to result in wider, more overlapped distributions. Second, objects with similar characteristics will result in distributions with more closely spaced expected values and tend towards more overlap. To assess the effect of the amount of characteristic overlap area on P_d, the difference between the expected values of the classes and the standard deviation of the measurements were varied for the three discrimination characteristics. The difference between the means was varied from 0.1 to 5 and the standard deviations were varied from 0.4 to 1. The amount of characteristic overlap was calculated for each combination of mean values and measurement standard deviation with 1000 trial runs for each combination.

Figure 21 depicts the variation of P_d with characteristic overlap area. An inverse quadratic relationship between characteristic overlap area and P_d was observed, with an r^2 of 0.9836 for the regression. As expected, as the amount of overlap area increased, the ability of the system to discriminate decreased. Further, this relationship was strongly quadratic, indicating that P_d decreases more rapidly as the overlap area increases. Conversely, decreasing the amount of overlap area will yield larger increases in P_d.

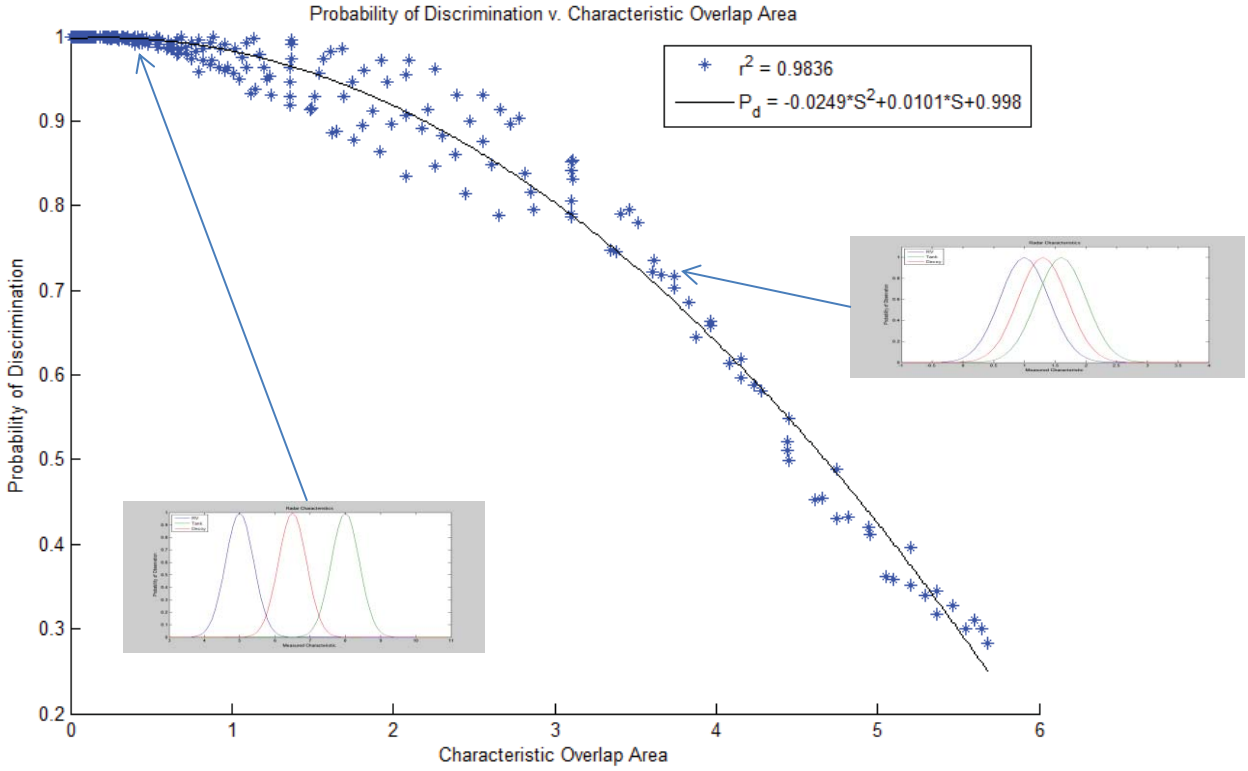

Figure 21. Variation of probability of discrimination with characteristic overlap area

6 Conclusions and Future Research

6.1 Conclusions

Using multiple, dissimilar sensors operated at different locations, this project developed an anchor Target Object Map (TOM) with least-squares based configuration selection for correlation and a Dempster-Shafer based discrimination algorithm for selecting the object of interest among closely-spaced objects travelling in a ballistic trajectory. A simulation environment was developed to conduct relevant operational scenarios and to assess the effectiveness of these algorithms with varying system parameters. This project found that varying +/-10% of the baseline radar range and angular resolution capability did not have a significant impact on the probability of discrimination (P_d). However, a 0.1 degree increase in angular resolution resulted in a 6 percent increase in the zero effort miss distance that increased the required KV maneuverability and optical sensor suite field of view of the KV at intercept. This project also found a strong inverse quadratic relationship between P_d and discrimination characteristic overlap, which is defined by the sensor measurement characteristics for different classes of objects. In order to achieve a P_d of 0.9, the characteristic overlap area could not exceed 2. Approaches to reduce characteristic overlap area should be explored to improve the discrimination performance of the system.

6.2 Future Research

The simulation capability developed in this project could be improved further to enhance the performance of the system via the inclusion of an Extended Kalman Filter[22] for tracking and a Mahalanobis distance correlation algorithm.[23] In addition, the fidelity of the simulation environment could be improved by modeling the KV optical sensor utilizing a pinhole optics model. These topics have been explored as stretch goals of the project but were not able to be completed in a timely manner. However, the ground work has been laid and can be employed to further enhance the multiple sensor discrimination capabilities.

6.2.1 Extended Kalman Filter State Estimation

Due to the nonlinear nature of orbital mechanics, the use of an Extended Kalman Filter could be beneficial in improving the state estimation performance. An Extended Kalman Filter takes the previously calculated states of the object and the next measurement and calculates the true states of the object at the given time step. The states can be calculated using Equation (11).[24]

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \bar{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{K}_{\boldsymbol{k}}[\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{h}(\bar{\boldsymbol{x}}_{\boldsymbol{k}})] \qquad (11)$$

In the preceding equation, $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$are the estimated states, $\bar{\boldsymbol{x}}_{\boldsymbol{k}}$ are the previous estimated states, $\boldsymbol{K}_{\boldsymbol{k}}$ are the Kalman gains for the particular measurement, z_k are the measured states, and $\boldsymbol{h}(\bar{\boldsymbol{x}}_{\boldsymbol{k}})$ are the expected measured states from the predicted states. $\bar{\boldsymbol{x}}_{\boldsymbol{k}}$ is found by integrating the nonlinear equations from the previous states to the next state. For example, Euler integration can

be utilized to propagate the states forward. The Kalman gains are found using the Riccatti equations, Equations (12), (13), and (14)[25],

$$\boldsymbol{M_k = \phi_k P_{k-1} \phi_k^T + Q_k} \quad (12)$$

$$\boldsymbol{K_k = M_k H^T (H M_k H^T + R_k)^{-1}} \quad (13)$$

$$\boldsymbol{P_k = (I - K_k H) M_k} \quad (14)$$

where $\boldsymbol{M_k}$ is the covariance matrix representing the errors in the state estimates prior to an update, and $\boldsymbol{P_k}$ is the covariance matrix following an update. **H** is the measurement matrix relating the measurements to states. $\boldsymbol{Q_k}$ is the process noise representing uncertainty in the behavior of the states, and $\boldsymbol{R_k}$ is the discrete measurement noise matrix representing the variances of each measurement source. $\boldsymbol{\phi_k}$ is the fundamental matrix, found via the Taylor-series expansion given in Equation (15), $\boldsymbol{F}$ is the systems dynamics matrix given by Equation (16) and T_s is the sampling time between measurements. It can be shown that the higher order terms can be neglected with minimal impact on the accuracy of the filter. Since the system is nonlinear, $\boldsymbol{F}$ is a first-order approximation to linearize the equations. [26]

$$\boldsymbol{\phi_k = I + F T_s + \frac{F^2 T_s^2}{2!} + \cdots \approx I + F T_s} \quad (15)$$

$$\boldsymbol{F = \frac{\partial f(x)}{\partial x}} \quad (16)$$

6.2.2 Mahalanobis Distance

Due to sensor uncertainty, the observed location of each object may differ between the two sensors. Mahalanobis distance can be used to calculate the correlation between the components of the target complex and is computed by Equation (17) [27], where r is the Mahalanobis distance from the feature vector $\underline{x}$ to the mean vector $\underline{m_x}$, and $\boldsymbol{C_x}$ is the covariance matrix of x.

$$r^2 = (\underline{x} - \underline{m_x})' \boldsymbol{C_x} (\underline{x} - \underline{m_x}) \quad (17)$$

Mahalanobis distance is utilized as it removes several limitations associated with using the Euclidean distance mainly that it accounts for the scaling of the coordinate axes and corrects for correlation between different object features.[28] The Mahalanobis distance between two tracks is denoted by $r_{m\text{-}n,}$, where m is the radar track number and n is the optical detection track number. The selected anchor objects will always be correlated. The remaining targets will be correlated based on the arrangement that minimizes Σr for the TOM.

6.2.3 Pinhole Optics Model

The optical sensor onboard the KV can be modeled using a simple pinhole optics model. The pinhole optics model involves defining a detector with a certain number of pixels and projecting this sensor plane through a "pinhole" to measure the range and bearing of the objects in the field of view (FOV). Each pixel can be assumed to have square dimensions. The sensor FOV was then divided by the number of pixels, known as the instantaneous FOV (iFOV) for each pixel. The FOV and iFOV of the sensor array will be varied to assess their impact on system performance.

---

[1] Jameson, Robert P., "Armageddon's Shortening Fuse: How Advances in Nuclear Weapons Technology Pushed Strategists to Mutually Assured Destruction, 1945-1962," *Air Power History,* Vol. 60, No. 1 (2013). (Accessed 30 April 2015).

[2] "The Threat," Missile Defense Agency, http://www.mda.mil/system/threat.html (accessed 30 April 2015).

[3] "MIM-104 Patriot," *Jane's Online* (2014). (Accessed 30 April 2015).

[4] Gompert, David C. and Jeffery A. Isaacson, "Planning a Ballistic Missile Defense System of System," *Rand Corporation IP-181* (1999). (Accessed 17 November 2014).

[5] Ibid.

[6] "The Ballistic Missile Defense System (BMDS)," Missile Defense Agency, http://www.mda.mil/system/system.html (accessed 30 April 2015).

[7] Gompert.

[8] Sanders-Reed, John N., "Multi-Target, Multi-Sensor, Closed Loop Tracking," SPIE 5340 (2004). (Accessed 30 April 2015).

[9] Garwin, Richard L., "Technical Aspects of Ballistic Missile Defense," *APS Forum on Physics and Society*, Vol. 28, No. 3 (1999). (Accessed 24 November 2014).

[10] Klein, Lawrence A, *Sensor and Data Fusion*, Bellingham, Washington: 2010. The Society of Photo-Optical Instrumentation Engineers.

[11] Sanders-Reed, J.N, "Multi-Target, Mult-Sensor, Closed Loop Tracking," Albuquerque, New Mexico: 2004, April. (Accessed 1 February 2014).

[12] Bate, Roger R., Donald D. Mueller, and Jerry E. White, *Fundamentals of Astrodynamics* (New York: Dover Publications, Inc., 1971), 86.

[13] Ibid, 80-84.

[14] Klein.

[15] Ibid.

[16] Silberman, Geoffrey L., "Parametric Classification Techniques for Theater Ballistic Missile Defense," *John Hopkins APL Technical Digest*, Vol. 19, No. 3 (1998). (Accessed 24 November 2014).

[17] Klein.

[18] Ibid.

[19] Ibid, 165.

[20] Ibid, 156.

[21] STK graphics generated utilizing an educational license.

[22] Zarchan, Paul and Howard Musoff, *Fundamentals of Kalman Filtering: A Practical Approach* (Reston, VA: American Institute of Aeronautics and Astronautics, 2009), 259.

[23] Mahalanobis Metric. (n.d.). (Princeton University) Retrieved October 27, 2014, from http://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_Mahal/M_metric.htm.

[24] Zarchan, 259.

[25] Ibid, 258.

[26] Ibid.

[27] Mahalanobis Metric.

[28] Ibid.

Bibliography

MIM-104 Patriot. (2014, June 26). (Jane's Online) Retrieved April 30, 2015, from https://janes.ihs.com/CustomPages/Janes/DisplayPage.aspx?DocType=Reference&ItemId=+++1501650&Pubabbrev=JAAD

The Ballistic Missile Defense System (BMDS). (2015, April 16). Retrieved April 30, 2015, from Missile Defense Agency: http://www.mda.mil/system/system.html

Bate, R. R., Mueller, D. D., & White, J. E. (1971). *Fundamentals of Astrodynamics.* New York: Dover Publications, Inc.

Garwin, R. L. (1999, July). Technical Aspects of Ballistic Missile Defense. *APS Forum on Physics and Society, 28*(3).

Gompert, D. C., & Isaacson, J. A. (1999). Planning a Ballistic Missile Defense System of Systems. *Rand Corporation IP-181*.

Jameson, R. P. (2013). Armageddon's Shortening Fuse: How Advances in Nuclear Weapons Technology Pushed Strategists to Mutually Assured Destruction, 1945-1962. *Air Power History, 60*(1). Rockville, Maryland. Retrieved April 30, 2015, from http://search.proquest.com/docview/1319809283?accountid=14748

Klein, L. A. (2010). *Sensor and Data Fusion.* Bellingham, Washington: The Society of Photo-Optical Instrumentation Engineers.

Ktasis, G. (2007, June). Multistage Security mechanism for Hybrid, Large Scale Wireless Sensor Networks. Monterey, California: Naval Postgraduate School. Retrieved April 12, 2015

Mahalanobis Metric. (n.d.). Retrieved October 27, 2014, from Princeton University.

Postol, T. A. (n.d.). Explanation of Why the Sensor in the Exoatmospheric Kill Vehicle Cannot Reliably Discriminate Decoys from Warheads. Retrieved October 30, 2014, from http://fas.org/spp/starwars/program/news00/postol_atta.pdf

Sanders-Reed, J. H. (2004, April). Multi-Target, Multi-Sensor, Closed Loop Tracking. *SPIE 5430*. Retrieved April 30, 2015

Silberman, G. L. (1998). Parametric Classification Technicques for Theater Ballistic Missile Defense. *John Hopkins APL Technical Digest, 19*(3).

Zarchan, P., & Musoff, H. (2009). *Fundamentals of Kalman Filtering: A Practical Approach.* Reston, VA: American Institute of Aeronautics and Astronautics.

Appendix I: Monte Carlo Simulation Coding

MIDN Samuel S. Lacinski, 25 April 2014

Project Abstract

This project examines the selection of an object of interest (OI) from a cluster of closely-space objects (CSOs) utilizing the returns from multiple sensors.

```
% This project is conducted in the SI system of units. Care is taken
% between sensor measurements, taken in units of meters, and orbital
% mechanics calculations, in units of kilometers.

% This program has five principle components: 1. Turth Target Complex
% Generation, 2. Radar Observation, 3. Optical Observation,
% 4. Target-Object Map (TOM) Correlation, and 5. Discrimiation Algorithm.
% In element will be explained in detail in the respective section of
% coding.
```

Workspace Preparation

```
clear
clc
format compact

% Declare Path
global currPath;
currPath = pwd;
path([currPath '/CoordinateTransforms/'],path)
path([currPath '/Discrimination/'],path)
path([currPath '/Filter/'],path)
path([currPath '/Sensors/'],path)
path([currPath '/TimeAndAngles/'],path)
path([currPath '/Trajectory/'],path)
path([currPath '/DataFiles/'],path)
path([currPath '/ECIFilter/'],path)
path([currPath '/SEZFilter/'],path)
path([currPath '/AddAxis/'],path)

global plotk;
plotk = [];              % plot index
```

Compute Truth Trajectories and Access Data

```
% Earth Paratemeters
global mu R_e w;
mu = 3.986004418e5;     % km^3/s^2, Earth Gravitational Parameter
R_e = 6371;             % km, Earth radius
```

```
w = [0;0;7.292115856e-5];   % rad/s, Earth angular rotation

% Target Complex Components
num_1 = 1;              % number of re-entry vehicles
num_2 = 1;              % number of tanks
num_3 = 2;              % number of spherical decoys
num_4 = 0;              % number of conical decoys
num_5 = 0;              % number of debris objects

target_comp = [num_1 num_2 num_3 num_4 num_5];
    % Target components vector
num_tar = sum(target_comp); % number of targets

% Define target parameters
global num_class;
num_class = 3;          % number of object classes

v_const = .01;

% global UTC_launch JD_launch file;
% file = 'PositionTrackData2.xlsx';         % data file

% Define CONOPs
global int_aq;
int_aq = 500;           % km, range of acquisition

% Define X1 position
radar_lat = 29.6837;               % deg, radar latitude
radar_long = -165.192;             % deg, radar longitude
radar_alt = 0;                     % km, radar altitude

X1.data = [radar_lat radar_long radar_alt];
```

Truth Target Complex

This section of code generates the ballistic trajectory of the components of the CSOs. This project is focused on the free-flight portion of the CSOs. This section of code also generates the true object charateristics. Each object has six total characteristics: a, b, c, alpha, beta, and gamma. a, b, and c are radar observable charateristics while alpha, beta, and gamma are optical observable charateristics.

The generated information is contained in a structured array. target contains a Nx1 cell matrix, where N is the true number of target elements. The trajectory element of each cell contains the julian day and corresponding J2000 position and velocity elements of the target elements.

```
% Compute trajectory of target elements
[complex,connect] = ScenarioInitial(target_comp,X1.data,v_const);
app = connect.app;
root = connect.root;
scenario = connect.scenario;
```

```
% Calculate discrimination characteristics and obtain epoch
in = 1;     % target index
for k = 1:num_class
    for n = 1:target_comp(k)
        if k == 1
            complex{in}.char = [1 4];
        elseif k == 2
            complex{in}.char = [3 6];
        else
            complex{in}.char = [2 5];
        end
        in = in+1;
    end
end

% Declare expected observation for target classes
global mean_char;
mean_char = [1 3 2;4 6 5];

% Determine initial launch time
global UTC_launch JD_launch;
UTC_launch = complex{1}.UTClaunch;
JD_launch = juliandate(UTC_launch);

% Import Access Data
X1.access = cell(num_tar,1);    % initialize access cell matrix

global JD_hitch
in = 1; % reset index
radar = connect.X1stk;     % select radar facility
for k = 1:num_tar % Loop over all complex objects
    % Compute access data for each object
    if k == 1
        Tar = connect.RVs;
    elseif k == 2
        Tar = connect.Tanks;
    elseif k == 3
        Tar = connect.Spheres;
    elseif k == 4
        Tar = connect.Cones;
    else
        Tar = connect.Debrs;
    end
    for n = 1:size(Tar,1)
        satid = Tar{n,1};   % object identifier
        sat = Tar{n,2};     % object
        % Compute Access
        access = radar.GetAccessToObject(sat);
        access.ComputeAccess;
        accessDP = access.DataProviders.Item('AER
Data').Group.Item('Default').Exec(scenario.starttime,scenario.stoptime,1);
        T = accessDP.DataSets.GetDataSetByName('Time').GetValues;
```

```
        T = UTC2Scenario(T);
        T = juliandate(T);
        az = cell2mat(accessDP.DataSets.GetDataSetByName('Azimuth').GetValues);
        el = cell2mat(accessDP.DataSets.GetDataSetByName('Elevation').GetValues);
        r = cell2mat(accessDP.DataSets.GetDataSetByName('Range').GetValues);
        azdot = cell2mat(accessDP.DataSets.GetDataSetByName('AzimuthRate').GetValues);
        eldot = cell2mat(accessDP.DataSets.GetDataSetByName('ElevationRate').GetValues);
        rdot = cell2mat(accessDP.DataSets.GetDataSetByName('RangeRate').GetValues);
        % Save data
        X1.access{in} = [T az el r azdot eldot rdot];
        % Find discontinunity
        JD_hitch(in,:) = FindAERDiscon(X1.access{in}(:,1),X1.access{in}(:,2));
        in = in+1;  % update index
    end
end

% Close STK
app.Quit();

% Load scenario values
load('Scenario.mat')
```

Declare Range of Variables for Iteration

Default Values: sigma_res = .0458 deg sigma_rng = 10/1e3 km

```
res = 0.44;                         % deg, baseline angular resolution
rng = 10/1e3;                       % km, baseline range resolution

sigma_res = [.9*res .95*res res 1.05*res 1.1*res];   % deg, angular resolution to be tested
sigma_rng = 1.05*rng;   % km, range resolution to be tested
```

Perform Iteration

This section of code implements a Monte Carlo simulation to assess the effect of the variation of the parameters of interest.

```
% Save globals into structured variable
globals.plotk = plotk;
globals.mu = mu;
globals.R_e = R_e;
globals.w = w;
globals.num_class = num_class;
globals.int_aq = int_aq;
globals.mean_char = mean_char;
globals.UTC_launch = UTC_launch;
globals.JD_launch = JD_launch;
globals.JD_hitch = JD_hitch;

loops = 300;        % number of iterations at each combination
```

```
%load('LastRun.mat')
n_last = 1;
m_last = 1;
results = zeros(loops,10);
P = zeros(length(sigma_res),10,length(sigma_rng));

for n = n_last:length(sigma_rng)
    if n == n_last
        m_loop = m_last;
    else
        m_loop = 1;
    end
    for m = m_loop:length(sigma_res)
        m
        n
        parfor k = 1:loops
            test = 0;
            iter = 0;   % number of attempts
            while test == 0 && iter < 10
                try
                    % Run simulation
                    results(k,:) = CSOMultiSensorDiscrimLoop(complex,X1,globals,...
                        sigma_res(m),sigma_rng(n));
                    test = 1;
                catch
                    % loop repeats
                    disp('Error detected. Repeating data point.')
                    iter = iter+1;
                end
            end
            if iter >= 10
                results(k,:) = NaN(1,8);
            end
        end
    P(m,:,n) = sum(results)/loops;
    save('LastRun.mat','m','n','P')
    end
end
```

Published with MATLAB® R2014a

Appendix II: Simulation Master Code

```
function [results] = CSOMultiSensorDiscrimLoop(complex,X1,globals,sigma_res,sigma_rng)
```

Declare Variables

```
global plotk mu R_e w num_class int_aq mean_char UTC_launch JD_launch JD_hitch
plotk = globals.plotk;
mu = globals.mu;
R_e = globals.R_e;
w = globals.w;
num_class = globals.num_class;
int_aq = globals.int_aq;
mean_char = globals.mean_char;
UTC_launch = globals.UTC_launch;
JD_launch = globals.JD_launch;
JD_hitch = globals.JD_hitch;
```

```
Error using CSOMultiSensorDiscrimLoop (line 4)
Not enough input arguments.
```

Create STK Scenario

```
% Create scenario
app = actxserver('STK10.Application'); % open STK
root = app.personality2;
scenario = root.Children.New('eScenario','Kwaj2Seattle');
% Set start and stop time
scenario.SetTimePeriod('1 Sep 2014 00:00:00','1 Sep 2014 01:00:00');
% Save to connect variable
connect.app = app;
connect.root = root;
connect.scenario = scenario;
```

Radar Observation

This section of code simulates the observation of the target complex by a radar system. The radar observes the target for a given period and then generates a TOM that is projected to the "eyes-open" time for the optical sensor. The return from the complex is the target position, velocity, and the radar signal return from the target.

This project assumed that the radar would add a random error to error element's truth location. This error is based on the radar system parameters. In addition, the radar signal return will have some random noise added to it based upon the internal noise of the system.

The radar simulated in this scenario is a nominal X-Band midcourse tracking and discrimination radar, termed the X1. The radar tracking parameters are stored in a structured array that contains the parameters of the system in .parameters and target tracks in .tracks.

```
% Define the radar system parameters
X1.sigma_rng = sigma_rng;       % km, range standard deviation
X1.ang_res = sigma_res;         % deg, angular resolution
X1.sigma_a = 0.5;               % characteristic a standard deviation
X1.PRF = 45;                    % Hz, pulse repetition frequency

X1.Pd = 1;                      % Probability of detection

X1.lat = X1.data(1);         % deg, radar latitude
X1.long = X1.data(2);        % deg, radar longitude
X1.alt = X1.data(3);         % km, radar altitude

% Generate target tracks
[X1,KV,complex,connect] = radarOBVwSTK(X1,complex,connect);
disp('Radar track')
```

Optical Observation

This section of code simulates the observation of the target complex by a space-based optical system. The sensor platform is also moving on a ballistic trajectory and observes the complex for "eyes open" time to the decision point. At the decision point, the observations are used to generate a TOM. The sensor observes the 2D position of the CSOs in the focal plane as well as characteristics of interest for discrimination purposes.

This project assumes the optical sensor would add a random error to each element's truth location, based on the system parameters. In addition, the observed characteristics would6 have an error resulting from the internal noise of the system.

```
% Define system parameters
KV.sigma_1 = 0.5;

% Observe target and generate TOM
KV = KVOBV_STK(X1,KV,complex);
disp('KV Acquired')
```

TOM Correlation

This section of code correlates the final TOMs generated by the radar and optical sensors. The correlation parameter generated is based upon the "2-D correlation" of the coordinates and measured velocities of the objects.

This project assumes no mismatches.

```
[optical_obv,TOM_data] = TOMCorr(X1,KV,complex);
if ~isempty(plotk)
    disp(['OV selected TOM configuration: ',num2str(TOM_data(2))])
    disp(['r = ',num2str(TOM_data(1),4)])
end
```

Discrimination Algorithm

This section of code fuses the discrimination information from the two sensors and produces a discrimination decision. The algorithm fuses the sensor inputs based upon the TOM correlation process in the previous section of code. Objects selected as matches by the correlation process will have the sensor returns matched in the object data base.

```
% Combine observations based on correlation process.
char_data = [X1.char KV.char(:,2:end)];

% Assign evidence measure based on each observation
num_obj = size(char_data,1);    % number of correlated objects
num_char = 2;                   % number of characteristics measured

char_sigma = [X1.sigma_a KV.sigma_1];

PPs = DempsterShaferOpen(mean_char,char_sigma,char_data);

% Identify Object of Interest

[F_imax,h] = max(PPs(:,1));
if ~isempty(plotk)
    disp(['The OI is ',num2str(h)])
    disp(['Interest factor is ',num2str(F_imax)])
end
```

Save Results

This section of coding determines if correlation, discrimination, and intercept were successful and saves information of interest.

```
% Determine if discrimination was successful (value of 1 indicates success)
if h == 1 && ~isnan(TOM_data(1))
    Disc_suc = 1;
else
    Disc_suc = 0;
end

% Determine if correct anchors were selected
if X1.anchor_id == 2 && ~isnan(TOM_data(1))
    radar_anc = 1;
else
    radar_anc = 0;
```

```
end
if KV.anchor_id == 2 && ~isnan(TOM_data(1))
    kv_anc = 1;
else
    kv_anc = 0;
end

% Determine if KV selected correct target independently
if KV.target == 1 && ~isnan(TOM_data(1))
    KV_int = 1;
else
    KV_int = 0;
end

% Determine if intercept was successful
if strcmp(complex{h,1}.ID,'RV') && ~isnan(TOM_data(1))
    Int_suc = 1;
else
    Int_suc = 0;
    TOM_data(1) = -1;
end

% Save results (format - [Corr Discrim Int Corr.coefficient PP_RV])
results = [TOM_data(3) Disc_suc Int_suc TOM_data(1) F_imax radar_anc kv_anc ...
    mean(X1.TU_min) KV_int KV.zem];
disp('Intercepted')
disp(' ')

% Close STK
if isempty(plotk)
    app.Quit();
end
```

Published with MATLAB® R2014a

Appendix III: Kalman Filter Code

```
function [Xk1,Pk1,Kk,TU,TU_v] = KalmanSEZ(X0,P0,Rk,zk,dt)
%KalmanSEZ performs one step of a linear Kalman filter on states in the SEZ
%frame.
%   The states are expected in the form [S,E,Z,Sdot,Edot,Zdot]', where
%   position is in km and velocity is in km/s. Pk is a 6x6 covariance
%   matrix in km^2. Rk is the measurement covariance matrix also in km^2.
%   Zk is the measurement matrix, expected as [S,E,Z], as velocity is
%   assumed to not be measured. If Zk is empty, the filter is propagated
%   without a measurement vector. T is the propagation time in seconds.

global mu R_e;

Rk = [Rk(1,1) 0 0;...
   0 Rk(2,2) 0;
   0 0 Rk(3,3)];

% State transition matrix
phi = eye(6,6);
phi(1,2) = dt;
phi(3,4) = dt;
phi(5,6) = dt;

% Input matrix
G = [dt^2/2 0 0;...
    dt 0 0;...
    0 dt^2/2 0;...
    0 dt 0;...
    0 0 dt^2/2;...
    0 0 dt];

% Process matrix
q = 1e-6;                           % Noise spectral density
Q = q*[dt^3/3 dt^2/2 0 0 0 0;...
    dt^2/2 dt 0 0 0 0;...
    0 0 dt^3/3 dt^2/2 0 0;...
    0 0 dt^2/2 dt 0 0;...
    0 0 0 0 dt^3/3 dt^2/2;...
    0 0 0 0 dt^2/2 dt];

% Measurement Matrix
H = zeros(3,6);
H(1,1) = 1;
H(2,3) = 1;
H(3,5) = 1;

% Gravitational Acceleration Matrix
g = -mu/norm([X0(1) X0(3) X0(5)+R_e])^2;    % estimated gravitational constant
u_k = [0;0;g];
```

```
% Best Estimate States
Xk = phi*X0+G*u_k;          % km, km/s, best estimate states
Pk = phi*P0*phi'+Q;         % km^2, best estimate covariance

% Update Estimates - omit if no measurement included
if ~isempty(zk)
    Kk = Pk*H'*inv(H*Pk*H'+Rk); % Kalman gain matrix
    Xk1 = Xk+Kk*(zk-H*Xk);      % km, km/s, updated estimates
    Pk1 = (eye(6,6)-Kk*H)*Pk;   % km^2, updated covariance matrix
else  % Use propagated estimates as updates
    Kk = NaN(6,3);
    Xk1 = Xk;
    Pk1 = Pk;
end

% Calculate Track Uncertainty at measurement instance
TU = norm([Pk1(1,1) Pk1(2,2) Pk1(3,3)]);
TU_v = norm([Pk1(4,4) Pk1(5,5) Pk1(6,6)]);

end
```

Published with MATLAB® R2014a

Appendix IV: MATLAB-STK Interface for Calculating Interceptor Trajectory

```
function [radar,KV,connect] = Intercept(radar,intercept_data,connect)

%Intercept This function determines the time of intercept for a given
%threat trajectory.
%   Output is in JD.
```

Variable Definiation

```
global int_aq;

num_tar = size(radar.trackSEZ,1);   % number of active tracks

% STK scenario data
app = connect.app;
root = connect.root;
scenario = connect.scenario;
```

```
Error using Intercept (line 10)
Not enough input arguments.
```

Select State Estimates

State estimates with the smallest average covariance normal for all tracked objects are utlized for determining the object orbit.

```
% % Determine number of loops
% loops = inf;
% for k = 1:size(radar.trackSEZ,1)
%     det = size(radar.trackSEZ{k,2},3);
%     if det < loops
%         loops = det;
%     end
% end
% ave = zeros(loops,1);
%
% % Loop over each track indicidence
% for k = 1:loops
% %    disp(['Loop: ',num2str(k)])
%     aves = zeros(num_tar,1);
%     % Average covariance for each track
%     for n = 1:num_tar
% %        disp(['Target: ',num2str(n)])
%         aves(n,1) = norm(radar.trackSEZ{n,2}(:,:,k));
%     end
%     ave(k,1) = norm(aves);  % km^2, average covariance for all tracks
% end
```

```
% Select point of minimum track uncertainty
h = zeros(1,num_tar);
radar.TU_min = zeros(num_tar,1);
for k = 1:num_tar
    [radar.TU_min(k),h(k)] = min(radar.trackSEZ{k,4}(:,3));
end

% Select states for orbit determination
%[~,h] = min(ave);
state_prop = zeros(num_tar,8);
for k = 1:num_tar
    statesSEZ = radar.trackSEZ{k,1}(h(k),:); % states for propagation
    % Convert states to ECI
    JD = statesSEZ(1,2);        % Julian day for state
    theta_LST = LST(JD,radar.long); % deg, local sidereal angle
    [r_ECI,v_ECI] = SEZ2ECI(statesSEZ(3:5)',statesSEZ(6:8)',radar.lat,...
        theta_LST);             % km, km/s, ECI states
    state_prop(k,:) = [statesSEZ(1:2) r_ECI' v_ECI'];
end
```

Project Trajectories

Using the selected state estimates, the tracks are projected forward until impact to determine the trajectory of the targets after tracking.

```
tars = cell(num_tar,2);         % initialize object handle matrix

root.ExecuteCommand('SetUnits / KM SEC UTCG');
for k = 1:num_tar
    tarid = ['TAR',num2str(k)];                         % target label
    tar = scenario.Children.New('eSatellite',tarid);    % target STK object
    % Initialize target with state estimates
    [year,month,day,hour,minu,sec,~,~] = julian2greg(state_prop(k,2));   % UTC time
    % Generate time string
    T = [year month day hour minu sec];
    radar.prop_start(k,1) = juliandate(T);
    T = UTC2Str(T);
    % Add object
    cmd = ['SetState */Satellite/',tarid,' Cartesian TwoBody ',T,' "',scenario.StopTime,...
        '" 5 J2000 ',T,' ',num2str(state_prop(k,3)),' ',num2str(state_prop(k,4)),...
        ' ',num2str(state_prop(k,5)),' ',num2str(state_prop(k,6)),...
        ' ',num2str(state_prop(k,7)),' ',num2str(state_prop(k,8))];
    root.ExecuteCommand(cmd);
    tars{k,1} = tarid;              % save object name
    tars{k,2} = tar;            % save object handle
    % Extract projected positions
    satDP = tar.DataProviders.Item('Cartesian
Position').Group.Item('J2000').Exec(scenario.starttime,scenario.stoptime,1);
    T = satDP.DataSets.GetDataSetByName('Time').GetValues;
    T = UTC2Scenario(T);
    T = juliandate(T);
```

```
    x = cell2mat(satDP.DataSets.GetDataSetByName('x').GetValues);
    y = cell2mat(satDP.DataSets.GetDataSetByName('y').GetValues);
    z = cell2mat(satDP.DataSets.GetDataSetByName('z').GetValues);
    satDP = tar.DataProviders.Item('Cartesian
Velocity').Group.Item('J2000').Exec(scenario.starttime,scenario.stoptime,1);
    xdot = cell2mat(satDP.DataSets.GetDataSetByName('x').GetValues);
    ydot = cell2mat(satDP.DataSets.GetDataSetByName('y').GetValues);
    zdot = cell2mat(satDP.DataSets.GetDataSetByName('z').GetValues);
    % Save data
    radar.prop_traj{k,1} = [T,x,y,z,xdot,ydot,zdot];
end
```

Determine Intercept Point

Based on the provided altitude in intercept_data, the function will determine the time and location of intercept.

```
int_alt = intercept_data.altitude;      % Desired intercept altitude
% Identify earliest time object breaks intercept altitude
T_int = inf;        % initialize intercept time
for k = 1:num_tar
    % Select object
    tarid = tars{k,1};
    tar = tars{k,2};
    % Compute altitude over flight
    tarDP = tar.DataProviders.Item('LLA
State').Group.Item('Fixed').Exec(scenario.starttime,scenario.stoptime,1);
    T = tarDP.DataSets.GetDataSetByName('Time').GetValues;
    T = UTC2Scenario(T);        % UTC
    T = juliandate(T);          % Julian Day
    alt = cell2mat(tarDP.DataSet.GetDataSetByName('Alt').GetValues);
    % Compute time at intercept altitude
    T_low = interp1(alt,T,int_alt); % Julian Day
    % Update intercept time, if necessary
    if T_low < T_int
        T_int = T_low;
    end
end

inter_rv = zeros(num_tar,7);
% Determine cartesion position and velocity at intercept time
for k = 1:num_tar
    % Select target
    tarid = tars{k,1};
    tar = tars{k,2};
    % Obtain Cartesian position and velocity
    satDP = tar.DataProviders.Item('Cartesian
Position').Group.Item('J2000').Exec(scenario.starttime,scenario.stoptime,1);
    T = satDP.DataSets.GetDataSetByName('Time').GetValues;
    T = UTC2Scenario(T);
    T = juliandate(T);
    x = cell2mat(satDP.DataSets.GetDataSetByName('x').GetValues);
```

```
    y = cell2mat(satDP.DataSets.GetDataSetByName('y').GetValues);
    z = cell2mat(satDP.DataSets.GetDataSetByName('z').GetValues);
    satDP = tar.DataProviders.Item('Cartesian
Velocity').Group.Item('J2000').Exec(scenario.starttime,scenario.stoptime,1);
    xdot = cell2mat(satDP.DataSets.GetDataSetByName('x').GetValues);
    ydot = cell2mat(satDP.DataSets.GetDataSetByName('y').GetValues);
    zdot = cell2mat(satDP.DataSets.GetDataSetByName('z').GetValues);
    % Interpolate position and velocity at intercept time
    X = interp1(T,x,T_int);
    Y = interp1(T,y,T_int);
    Z = interp1(T,z,T_int);
    Xdot = interp1(T,xdot,T_int);
    Ydot = interp1(T,ydot,T_int);
    Zdot = interp1(T,zdot,T_int);
    % Save data
    inter_rv(k,:) = [T_int X Y Z Xdot Ydot Zdot];
end

% Determine aim point (aim point taken as average of all objects position
% at intercept time)
aim_point = [T_int mean(inter_rv(:,2)) mean(inter_rv(:,3)) mean(inter_rv(:,4))];
KV.aim_point = aim_point;
```

Determine Interceptor Trajectory

This segment of code calculates the trajectory necessary for the interceptor to converge on the target.

```
% Create interceptor object
KV1 = scenario.Children.New('eSatellite','KV1');

% Create target sequence and remove default initial state and propagat
root.ExecuteCommand('Astrogator */Satellite/KV1 InsertSegment
MainSequence.SegmentList.Initial_State Target_Sequence');
root.ExecuteCommand('Astrogator */Satellite/KV1 DeleteSegment
MainSequence.SegmentList.Initial_State Target_Sequence');
root.ExecuteCommand('Astrogator */Satellite/KV1 DeleteSegment MainSequence.SegmentList.Propagate
Target_Sequence');

% Create launch, manuever, and propagate sequence
root.ExecuteCommand('Astrogator */Satellite/KV1 InsertSegment
MainSequence.SegmentList.Target_Sequence.SegmentList.- Propagate');
root.ExecuteCommand('Astrogator */Satellite/KV1 InsertSegment
MainSequence.SegmentList.Target_Sequence.SegmentList.Propagate Launch');
root.ExecuteCommand('Astrogator */Satellite/KV1 InsertSegment
MainSequence.SegmentList.Target_Sequence.SegmentList.Propagate Maneuver');

% Set constants for launch
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.TimeOfFlight 63 sec');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Burnout.FixedVelocity 8.3 km/sec');
```

```
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Launch.Geodetic.Latitude 34.7561
deg');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Launch.Geodetic.Longitude -120.626
deg');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Launch.Geodetic.Altitude 0.01 km');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Maneuver.ImpulsiveMnvr.AttitudeControl
Thrust Vector');

% Configure target sequence control parameters
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch Burnout.LaunchAzDRDAlt.Altitude');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Launch
Burnout.LaunchAzDRDAlt.Altitude Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Burnout.LaunchAzDRDAlt.Altitude 225
km');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch
Burnout.LaunchAzDRDAlt.DownrangeDistance');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Burnout.LaunchAzDRDAlt.DownrangeDista
nce 399.399 km');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Launch
Burnout.LaunchAzDRDAlt.DownrangeDistance Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch Burnout.LaunchAzDRDAlt.LaunchAZ');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Burnout.LaunchAzDRDAlt.LaunchAz -
17.8115 deg');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Launch
Burnout.LaunchAzDRDAlt.LaunchAZ Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch Launch.Epoch');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Launch.Launch.Epoch "1 Sep 2014 00:23:00"
UTCG');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Launch Launch.Epoch
Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Maneuver ImpulsiveMnvr.Cartesian.X');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Maneuver
ImpulsiveMnvr.Cartesian.X Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Maneuver ImpulsiveMnvr.Cartesian.Y');
```

```
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Maneuver
ImpulsiveMnvr.Cartesian.Y Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Maneuver ImpulsiveMnvr.Cartesian.Z');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Maneuver
ImpulsiveMnvr.Cartesian.Z Active true');
root.ExecuteCommand('Astrogator */Satellite/KV1 AddMCSSegmentControl
MainSequence.SegmentList.Target_Sequence.SegmentList.Propagate
StoppingConditions.Duration.TripValue');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Propagate.StoppingConditions.Duration.TripVa
lue 135.336 sec');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSControlValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate
StoppingConditions.Duration.TripValue Active true');

% Configure constraints
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.SegmentList.Propagate.Results Epoch X Y Z');
root.ExecuteCommand(['Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate X Desired
',...
    num2str(aim_point(2)*1e3)]);
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate X Active
true');
root.ExecuteCommand(['Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate Y Desired
',...
    num2str(aim_point(3)*1e3)]);
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate Y Active
true');
root.ExecuteCommand(['Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate Z Desired
',...
    num2str(aim_point(4)*1e3)]);
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate Z Active
true');
[year,month,day,hour,minu,sec,~,~] = julian2greg(T_int);
T_intUTC = [year month day hour minu sec];      % UTC intercept time
T_intUTC = UTC2Str(T_intUTC);
root.ExecuteCommand(['Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate Epoch Desired
',...
    T_intUTC,' UTCG']);
root.ExecuteCommand('Astrogator */Satellite/KV1 SetMCSConstraintValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector Propagate Epoch Active
true');
```

```
% Initialize corrector sequence
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.Profiles.Differential_Corrector.Mode Iterate');
root.ExecuteCommand('Astrogator */Satellite/KV1 SetValue
MainSequence.SegmentList.Target_Sequence.Action Run active profiles');
% Run MCS sequence
root.ExecuteCommand('Astrogator */Satellite/KV1 RunMCS');
root.ExecuteCommand('Astrogator */Satellite/KV1 ApplyCorrections
MainSequence.SegmentList.Target_Sequence');
root.ExecuteCommand('Astrogator */Satellite/KV1 ClearDWCGraphics');

root.ExecuteCommand('Animate * Reset');

% Extract position elements
KVDP = KV1.DataProviders.Item('Cartesian
Position').Group.Item('J2000').Exec(scenario.starttime,scenario.stoptime,1);
T = KVDP.DataSets.GetDataSetByName('Time').GetValues;
T = juliandate(T);
x = cell2mat(KVDP.DataSets.GetDataSetByName('x').GetValues);
y = cell2mat(KVDP.DataSets.GetDataSetByName('y').GetValues);
z = cell2mat(KVDP.DataSets.GetDataSetByName('z').GetValues);
KVDP = KV1.DataProviders.Item('Cartesian
Velocity').Group.Item('J2000').Exec(scenario.starttime,scenario.stoptime,1);
xdot = cell2mat(KVDP.DataSets.GetDataSetByName('x').GetValues);
ydot = cell2mat(KVDP.DataSets.GetDataSetByName('y').GetValues);
zdot = cell2mat(KVDP.DataSets.GetDataSetByName('z').GetValues);
% Save Data
KV.trajectory = [T,x,y,z,xdot,ydot,zdot];
```

Determine Intercept Acquisition Time

This step calculates the time at which the KV acquires the target based upon the global intercept range variable.

```
r = inf;        % initialize range to target
in = 1;         % loop index
while r > int_aq
    R_KV = KV.trajectory(in,2:4);
    R = aim_point(2:4)-R_KV;
    r = norm(R);
    in = in+1;
end

KV.JD_int = KV.trajectory(in,1);        % Julian Day, time at acquisition
[year,month,day,hour,minu,sec,~,~] = julian2greg(KV.JD_int);
KV.UTC_int = [year month day hour minu sec];

end
```

Published with MATLAB® R2014a

Appendix V: TOM Correlation Algorithm Script

```
function [KV,TOM_data] = TOMCorr(radar,KV,complex)

%TOMCorr processes all the possible combinations of TOMs and selects the
%arrangement with the most likely probability.
%   This function cannot handle mismatches. If a plot_in (plot index) is
%   provided, this function will plot all the possible TOMs on a subplot.
```

Variable Definition

```
global plotk;

% Subplot Dimensions
N = size(KV.TOM,1);
a = ceil(N/2);
```

TOM Correlation

Correlation is based strictly on the geometric position of the objects in the focal plane.

```
% % Calculate azimuth and elevation from radar TOMs
% [~,radar_ang] = rangeangle(radar.TOM(:,2:4)');  % rad
%     % output in form [azimuth;elevation]
% radar_TOM = [(1:N)' deg2rad(radar_ang(1,:))' deg2rad(radar_ang(2,:))'];
%     % rad

radar_TOM = radar.TOM2;

% All possible optical TOM arrangements
optical_TOMs = cell(N,1);

optical_TOM = KV.TOM;
optical_TOMIDs = cell(N,1);
optical_TOMID = KV.TOMID;
TOM_actual = KV.TOM;

r = zeros(N,1);

for m = 1:N
    % Save previous optical TOM
    prev_optical = optical_TOM;
    prev_ID = optical_TOMID;

    % Modify previous radar TOM to next arrangement
    if m > 1
        for q = 1:N
            in = q+1;   % index for moving TOM entries
            if in > size(prev_optical,1)
```

```
                % correct for index too large
                in = in-size(prev_optical,1);
            end
            optical_TOM(q,:) = prev_optical(in,:);
            optical_TOMID{q,:} = prev_ID{in,1};
        end
    end

    % Save optical TOM arrangement
    optical_TOMs{m} = optical_TOM;
    optical_TOMIDs{m} = optical_TOMID;

    % Calculate Correlation for particular optical TOM arrrangement
    r(m) = corr2(radar_TOM(:,2:3),optical_TOMs{m}(:,2:3));

    % Plot Each Correlation
    if ~isempty(plotk)
        figure(plotk)
        if m == 1
            clf
        end
        subplot(a,2,m), hold on
        h1 = plot(radar_TOM(:,2),radar_TOM(:,3),'vb');
        h2 = plot(optical_TOM(:,2),optical_TOM(:,3),'^r');
        for k = 1:size(radar_TOM,1)
            plot([radar_TOM(k,2) optical_TOM(k,2)],[radar_TOM(k,3),...
                optical_TOM(k,3)],'Color',[.2 k/size(radar_TOM,1) 0]);
        end
        title(['TOM ',num2str(m),', r = ',num2str(r(m),4)])
        xlabel('Azimuth (rad)')
        ylabel('Elevation (rad)')
        axis equal
        %legend([h1,h2],'Location','EastOutside','Radar Targets',...
        %'Optical Targets')
        % Label Targets
        space = mean(mean([radar_TOM(:,2)-optical_TOM(:,2),radar_TOM(:,3)-...
            optical_TOM(:,3)]));  % Calculate normalized spacing
        for p = 1:N
            text(radar_TOM(p,2)+space,radar_TOM(p,3)+space,num2str(radar_TOM(p,1)))
            text(optical_TOM(p,2)-space,optical_TOM(p,3)-space,num2str(p))
        end
    end
end
% Create additional plot space if needed
% if m > a*2
%     a = a+1;
% end
% subplot(a,2,m+1), hold on

if ~isempty(plotk)
    plotk = plotk+1;
    legend([h1 h2],'Location','Best','Radar Targets','Optical Targets')
end
```

```
% Select TOM with best correlation

% r_max is the max correlation
% h is the index for the max correlation
[r_max,h] = max(r);
TOM_data = [r_max,h,0];   % Output correlation and TOM configuration

% Select optical TOM of particular arrangment
KV.TOM = optical_TOMs{h};
KV.TOMID = optical_TOMIDs{h};

% Re-order optical charateristic observations
% Original data
optical_char0 = KV.char;
m0 = h-1;               % shift required

for k = 1:N
    m = m0;
    if (k+m) > N
        m = m-N;
    end
    % Update optical observations
    KV.char(k,:) = optical_char0(k+m,:);
end
% Determine if correlation was successful
space = max(radar_TOM(:,2))/10;
suc = 1;
for k = 1:N
%     if ~strcmp(KV.TOMID{k,1},complex{k,1}.ID)
%         suc = 0;
%     end
    if KV.TOM(k,1) ~= k
        suc = 0;
    end
end
% Plot TOM
if ~isempty(plotk)
    figure(plotk)
    clf; hold on
    for k = 1:N
        plot(radar_TOM(k,2),radar_TOM(k,3),'^r')
        plot(KV.TOM(k,2),KV.TOM(k,3),'vb')
        text(radar_TOM(k,2)+space,radar_TOM(k,3)+space,...
            num2str(radar_TOM(k,1)))
        text(TOM_actual(k,2)-space,TOM_actual(k,3)-space,...
            ['A',num2str(TOM_actual(k,1))])
        plot([radar_TOM(k,2) KV.TOM(k,2)],[radar_TOM(k,3) KV.TOM(k,3)],'-',...
            'Color',[.2 k/size(radar_TOM,1) 0]);
        xlabel('Azimuth (rad)')
        ylabel('Elevation (rad)')
        title(['Selected TOM Configuration, r = ',num2str(r_max,3)])
        plotk = plotk+1;
    end
end
```

```
TOM_data(3) = suc;

end
```

Published with MATLAB® R2014a

Appendix VI: Dempster-Shafer Discrimination Algorithm Coding

The code for the Dempster-Shafer algorithm was developed by Dr. Sami Amashi, MIT-Lincoln Laboratory, Group 31.

```
function [PPs] = DempsterShaferOpen(means,sigmas,measures)
%DempsterShaferOpen calculates the pignistic probabilities of a given
%series of M measurements belonging to each class given a set of observations
%and the expected distributions of those measurements for each class. It
%utilizes an open-world assumption and does not eliminate any conflict in
%the measurements. N is the number of objects observed
%   Inputs: means   - expected observations, expected as a MxN row vector
%           sigmas  - observation standard deviations as a 1xN row vector
%           obv     - measurements expected as an Mx(N+1) matrix, where
%                     the first row is object indices.
%
%   Outputs:    PPs - Pignistic Probabilities for each class.
%
%   The Dempster-Shafer functionality of this program was writen by Sami
%   Amashi, MIT-Lincoln Laboratories. This function merely assembles his
%   functions into one function.

global num_class;

N = size(measures,1);       % Number of objects
M = size(means,1);          % Number of characteristics

PPs = zeros(N,num_class);

for k = 1:N % Loop over each detection
    for m = 2:(M+1) % Loop over each charateristic
        obv = measures(k,m);   % observed characteristic
        % Compute probablity of each measurement from each distribution
        mus = means(m-1,:);     % expected observations
        sigma = sigmas(m-1);    % observation SD
        [pb1,pb2,pb3] = ProbAssign3(obv,mus,sigma);
        class_prob = [pb1;pb2;pb3]; % probability of each observation from each class
        % Assign evidence measure
        EM = DS_Consonant_UnNormalized(class_prob);
        % Combine previous and current evidence measures
        if m > 2 % skip combination for first characteristic
            EM_prev = DS_OWCombination(EM,EM_prev);
        else
            EM_prev = EM;
        end
    end
    % Calculate pignistic probabilities based on open world
    PPs(k,:) = DS_PignisticProbability(EM_prev,1)';
end
```

```
end
```

Published with MATLAB® R2014a